sinclair ZX SPECTRUM BASIC PROGRAMMING



מדריך תכנות בסיסי בשפת ה- BASIC המורחבת של 'סינקלייר'

סטיבן ויקרס ובראדביר

מדריך תכנות בסיסי
בשפת ה־BASIC המורחבת של סינקלייר

סטיבן ויקרס ורובין בראדביר

אידאה בע"מ
הוצאה לאור

sinclair ZX SPECTRUM BASIC PROGRAMMING

by Steven Vickers edited by Robin Bradbeer

IDEA L.T.D, Publishers
P.O.B. 180 GIVATAIM, ISRAEL

תרגום: זיכלנסקי אריה
עיצוב עטיפה: רות וון דה גרדה

הוצאה לאור אידאה בע"מ
ת.ד. 180, גבעתיים

נדפס בישראל תשמ"ג/1983

תוכן עניינים

המחשב והפעלתו הראשונית

מבוא זה נכתב עבור אנשים שאינם יודעים מאומה על מחשבים ועבור אלו שמערכות מחשבים מוכרות להם אולם הם מעונינים לקרוא חוברת הוראות לפני ההפעלה הראשונית. בהמשך יופיע המדריך לתכנות ב־BASIC, על המתחיל לקרוא את הספר אך ורק לאחר שקרא והבין מבוא זה.

כשתפרק את אריזת ה־Zx spectrum תמצא:

1) מבוא ומדריך לתכנית ב־BASIC.

2) מחשב שבגבו שלושה שקעים קטנים המסומנים 9V DC IN, EAR ו־MIC שקע טלויזיה ו"מחבר קצה" (edge connector) אליו ניתן לחבר ציוד נוסף. כדי להפעיל את המחשב צריך לחברו לספק הכוח בלבד.

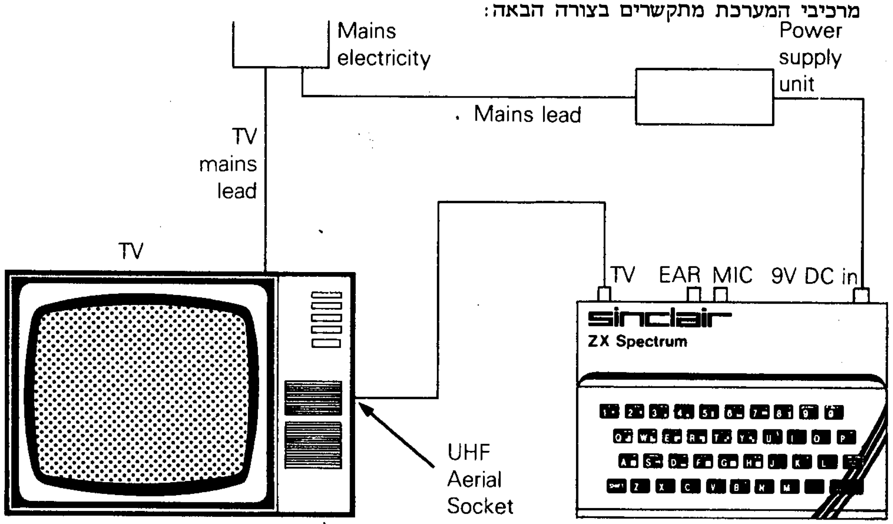
3) ספק הכוח — ממיר את מתח רשת החשמל הארצית למתח שה־ZX Spectrum משתמש. אם תרצה להשתמש במקור אספקת כוח עצמאי משלך עליו לספק 9V זרם ישר בזרם של 1.4A בלתי מיוצב.

4) חוט אנטנה כאורך של כ־2 מטר המחבר את המחשב לטלויזיה.

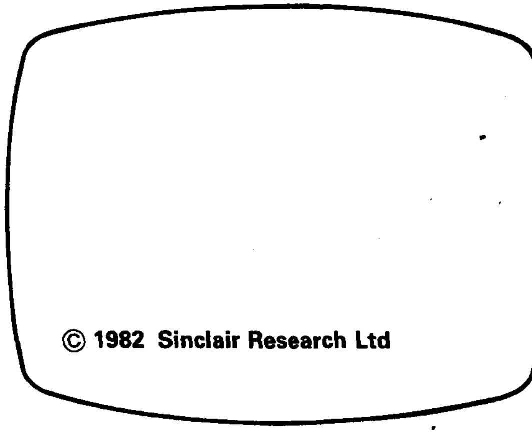
5) זוג חוטים באורך של כ־75 ס"מ בעלי תקעים של 3.5 מ"מ בכל קצה המחברים את המחשב לרשם הקול.

תצטרך גם טלויזיה — ה־Zx spectrum יכול לפעול בלעדיה אבל לא תוכל לראות מה הוא מבצע. הטלויזיה צריכה להיות בעלת ערוצי U.H.F (רוב הטלויזיות המוכרות בארץ מתאימות למחשב זה) ה־Zx spectrum כשמו כן הוא הוא מיצר אותות צבעוניים שבעזרת מקלט טלויזיה צבעוני יתנו תמונה צבעונית. אם תשתמש במכשיר טלויזיה שחור לבן, הצבעים יופיע בשחור לבן וששה גוונים של אפור, להוציא את ענין הצבע, מכשיר טלויזיה שחור לבן יתפקד היטב, באותה מידה כמו טלויזיה צבעונית.

מרכיבי המערכת מתקשרים בצורה הבאה:



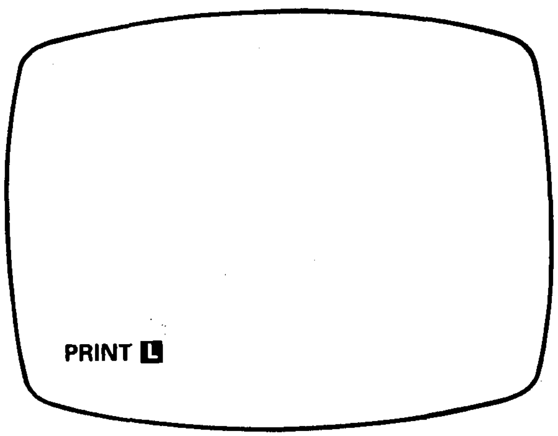
תמונה 1
אם בטלויזיה ישנם שני שקעי אנטנות המסומנים UHF ו־VHF יש להשתמש בשקע המסומן UHF. הפעל את המחשב והטלויזיה וכוון את הטלויזיה לערוץ 36 בְּ־UHF כאשר המחשב מופעל והטלויזיה מכוונת תתקבל התמונה הבאה:



תמונה 2
כאשר תשתמש במחשב בודאי תרצה להפחית את עוצמת הקול בטלויזיה עד למינימום.
אם למכשיר הטלויזיה יש כפתורי הזזה לכוונון ובקרה רצופה תוכל להתאימם עד שתקבל את תמונה 2 על המסך. טלויזיות רבות כיום הינן בעלות לחצנים עבור כל תחנה. בחר לחצן שאיננו בשימוש וכוון אותו לערוץ 36 UHF.
כדי להשתמש במחשב בְּארצות בעלות שיטת טלויזיה שונה מזו הנהוגה בארץ דרושה גירסה שונה של ה־Zx spectrum שתוכננה במיוחד לאותה שיטה. גירסת המחשב בארץ משתמשת בערוץ UHF בעל 625 שורות ובהקרנה של המסך 50 פעם בשניה ובשיטת איפנון צבע הנקראת PAL.
כשתכבה את ה־Zx spectrum כל המידע שאוחסן בו יאבד. אחת הדרכים לשמור מידע זה לשימוש מאוחר יותר היא ע״י הקלטתו על קסטה של רשם קול. תוכל גם לקנות קסטות שאנשים אחרים הכינו ולהריץ את התוכניות שלהם.
החוט בעל 2 התקעים בכל קצה משמש לחיבור ה־Zx spectrum לרשם הקול הביתי הסטנדרטי.
אחרי שהפעלת את המחשב, תרצה בוודאי להשתמש בו וזאת תלמד בהמשך החוברת. אם מחוסר סבלנות התחלת ללחוץ על המקשים תגלה שפעולה זו מסירה את סימן הודעת זכויות היוצר, לא תוכל לפגוע במחשב בצורה זו, העז ונסה ללמוד ואם תתקע תוכל תמיד להחזיר את המחשב לתמונה המקורית ע״י הוצאת התקע המסומן 9V DC IN מהמחשב ולהכניסו בחזרה. השתמש בשיטה זו באפשרות האחרונה כי כך אתה מאבד את כל המידע שבמחשב.
אזהרה אל תנסה לחבר את ה־ZX 16K RAM ל־Zx spectrum זה לא יפעל.

לוח המקשים

לוח המקשים של הספקטרום דומה למכונת כתיבה סטנדרטית. האותיות והמספרים נמצאים באותם מקומות אלא שכל מקש יכול לבצע מספר פעולות. במכונת כתיבה רגילה מופיעות אותיות קטנות ובשימוש של מקש ה־shift מופיעות אותיות גדולות (capitals) לוח המקשים של הספקטרום פועל באותה שיטה בדיוק.
כדי לעזור לך לדעת באיזה מצב לוח המקשים נמצא, מופיעה אות לבנה על רקע שחור על מסך הטלויזיה. אות זו נקראת ה״מצביע״ (Cursor) כאשר הספקטרום מופעל הוא מראה את הודעת זכויות היוצרים על גבי הצג. לחיצה על כל מקש תגרום למלה המודפסת מתחת לאות שבמקש להופיע. מלה זו נקראת מלת מפתח. מאחר והמחשב מצפה לפקודה שתאמר לו מה לבצע, כל הפקודות חייבות להתחיל עם מלת מפתח. הספקטרום מאפשר לך להכניס מלת מפתח בלחיצת מקש אחת בלבד (שלא בדומה לרוב המחשבים האחרים). לדוגמה אם נקיש על P מיידית לאחר הפעלת המחשב תופיע על המחשב מלת המפתח PRINT; ״הסימן״ מסומן גם הוא במקש האות P. כדי לקבלו עליך ללחוץ על שני מקשים בבת אחת. לחץ בנגיעה על מקש ה־SYMBOL SHIFT שנמצא בסמוך לפנה הימנית תחתונה של לוח המקשים ובו זמנית לחץ על מקש ה־P.



תמונה 3
המצביע עכשיו משתנה ל־L והמחשב מצפה לאות (Letter) הדפס את האותיות ״Hello״, אם נמצא טקסט אחר על המסך כבה את המחשב (ע״י הוצאת תקע ה־9V) והתחל מחדש. השתמש במקש ה־CAPS SHIFT כדי לקבל את האות הגדולה H, באופן כללי כל דבר המסומן לבן על המקש מצריך שימוש ב־CHPS SHIFT וכל דבר הכתוב באדום על גבי המקשים מצריך שימוש ב־SYMBOL SHIPT פקודה המתחילה בהוראה PRINT מורה למחשב לרשום על המסך את האותיות המסומנות בגרשיים (מרכאות כפולות), כדי לגרום למחשב לבצע את ההוראה יש ללחוץ על מקש ENTER, כשפעולות אלה בוצעו הצג צריך להראות את המלה: HELLO

ומספר אותיות נוספות. סימן שאלה מהבהב מסמל הודעת־שגיאה ובמקרה שהוא מופיע התחל מחדש וחזור על התרגיל. ההודעה בתחתית המסך היא באמת הודעת המחשב כי ההוראה היא נכונה ('OK') ההודעה חשובה כשמריצים תוכניות אך ניתן להתעלם ממנה כרגע.

שים לב לדבר נוסף: האות O (או) והמספר Ø (אפס) מיוצגים ע"י צורה שונה וחשוב לזכור זאת. דרך המספר Øתמיד עובר קו, והמחשב תמיד יתיחס לאות O כאות, לכן יש לשים לב ולא ללחוץ על המקש הלא נכון. באופן דומה המספר 1 והאות L קטנה (l) שונים. שלא בדומה למספר מכונות כתיבה אין אפשרות לשנות סימן זה.

מכיוון שחשוב להבין את מצבי לוח המקשים יהיה זה שימושי להסביר מה קורה פעם נוספת. הסימן המהבהב L נקרא — המצביע והוא מראה היכן על הטלויזיה ישים המחשב את הפריט הבא שתדפיס. אם תכבה ותדליק את המחשב סימן זכויות היוצרים ישתנה למצביע מסוג K.

האות שבתוך המצביע מראה איך יתיחס המחשב לפריט הבא שתדפיס, בתחילת שורה יהיה זה K מהבהב שפרושו keyword — מלת מפתח. סימן ה־C והודעות אחרות נחשבות גם הן לK המהבהב.

מלת מפתח היא אחת ממלות המפתח המיוחדות של המחשב המופיעות בתחילת כל פקודה ומבהירות למחשב מה עליו לבצע. מכיוון שהמחשב מצפה למילת מפתח בתחילת השורה כאשר תלחץ על מקש ה־P המחשב יחליט לא להדפיס P אלא PRINT והוא מודיע לך ע"י ה־K שזאת תהיה הפעולה שהוא יבצע. כאשר מתקבלת מלת מפתח אחת לא תתקבל אחרת ולכן כל מה שתדפיס עכשיו יתקבל כאותיות. כדי להודיע לך על כך המחשב משנה את המצביע ל־L שפרושו אות ('Letter').

סוגים שונים אלה נקראים לרוב "מצבים" ולהבא נדבר על מצב מלות המפתח K או מצב האותיות L. אם תרצה להדפיס מספר גדול של אותיות מבלי ללחוץ באופן קבוע על ה־CAPS SHIFT תוכל לגרום לכל האותיות להופיע גדולות ע"י לחיצה CAPS LOCK (CHPS SHIFT ואז 2) כדי להראות שזה קורה, המצביע L יתחלף ב־C מהבהב (שפרושו אותיות גדולות Capitals). כדי לקבל אותיות קטנות ומצביע L לחץ שנית על CAPS LOCK, אם תלחץ על CAPS LOCK בזמן מצב מלות המפתח K לא תבחין מיד בשינוי אבל תגלה את השינוי לאחר הכנסת מלת מפתח כאשר המחשב יהיה במצב C במקום ב־L.

בנוסף למילות מפתח ושני סוגי האותיות בלוח המקשים ישנם גם שמונה סימנים גרפיים המופיעים על מקשי המספרים 1 עד 8 וניתן להדפיסם על הצג בצורה דומה לאותיות ומספרים.

כדי לבצע זאת יש לשנות את מצב לוח המקשים למצב גרפי שינוי זה מבוצע ע"י לחיצת מקש ה־CAPS SHIFT עם ה־9. שים לב שהמצביע משתנה ל־G. לחיצה על 9 תחזירו שוב למצב L.

ישנו מצב נוסף שניתן לשנות את לוח המקשים אליו, והוא מצב ההרחבה (Extended) שמסומן ב־E. הוא מושג ע"י לחיצת CAPS SHIFT וגם SYMBOL SHIFT בו זמנית מצב זה מאפשר להשתמש ברוב הפונקציות המדעיות והתכנותיות לחיצת שני הכפתורים שוב תחזיר את לוח המקשים למצב אותיות L.

גם הכתבנית והמתכנת המקצועיים ביותר טועים מדי פעם בלחיצת המקשים. למדנו עד כה לתקן טעויות ע"י כיבוי המחשב. שיטה זו נוחה אם רק פקודה אחת ניתנה למחשב אבל ללא ספק בלתי יעילה אם כבר הוכנס מידע רב.

למזלנו ניתן להשתמש במקש ה־DELETE למחוק שגיאות. כדוגמא אפשרית אפילו טעות בפקודה הפשוטה:

PRINT "Hello"

נניח שלא השתמשת SYMBOL SHIFT כדי לקבל את הגרשיים הפותחים ואז הצג יראה:

PRINT "PHELLO"

המחשב לא יוכל לזהות מה שבא לאחר ה־PRINT מכיוון שללא הגרשיים המחשב מצפה למספר ובמקום הספרה נמצאת אות. המחשב מראה את מבוכתו ע"י הבהוב של סימן ה־? בסוף השורה. למזלנו אין צורך להדפיס את כל השורה שנית. בשורה הראשונה של לוח המקשים ישנם ארבעה חיצים המצביעים לכיוונים שונים והמילה DELETE (מחק). להפעלת מקשים אלה עליך ללחוץ על CAPS SHIFT בו זמנית עם לחיצה על אחד מהם.

החיצים שבצדדים מזיזים את המצביע ימינה או שמאלה ומקש DELETE מוחק את הסימן שבא מיד לפני המצביע.

כדי לתקן את השורה השגויה לחץ על (CAPS SHIFT וְ־5 בו זמנית) עד שהמצביע נמצא מיד לאחר ה־P. השגוי. אם תלחץ את שני המקשים לשניה או שתים הם יתחילו לעבוד באופן רצוף וישמיעו קליק חלש. למעשה לחיצה על מקש ליותר משתי שניות תגרום לחזרה אוטומטית על הפעולה. לחץ על DELETE (CAES SHIFT וְ Ø) בכדי למחוק את ה־P השגוי ואז הדפס (גרשיים — SYMBOL SHIFT וְ־P).

שים לב שכל שורה נדחפה ימינה מבלי לבצע שום פעולה אחרת. נסה את החץ שמסמל 'מצביע ימינה' רק כדי לראות איך זה עובד. אם בצעת שגיאות תקנן באותה דרך. זכור ששגיאות תוכל למחוק ולהכניס במקומן תיקונים.

עכשיו לחץ על ENTER, המחשב יכתוב את הודעתך בראש המסך או מתחת להודעה שנכתבה לפני כן אך היא עדיין קיימת.

תאור מלא של לוח המקשים תמצא בפרק 1 של מדריך התכנות ב־BASIC

מספרים, אותיות והמחשב במחשב כיס

עד כה ראינו איך לפקוד על המחשב שידפיס אותיות וסימנים גרפיים ע"י שימוש בפקודה PRINT. כמו כן ראינו כי יש להשתמש במקש ENTER על מנת לומר למחשב שיבצע את הפקודה שהודפסה. מעבשיו לא יוזכר כי יש ללחוץ על ENTER ומובן מאליו שתקיש אותו בראש כל שורה באופן אוטומטי.

המחשב יכול לטפל במספרים יותר בקלות מאשר באותיות בפרק הקודם רמזנו שהמחשב מצפה למספר לאחר הפקודה PRINT כאשר אין משתמשים בגרשיים כך שאם נקיש

```
PRINT 2
```

המספר 2 יופיע על המסך.
בהחלט ניתן לערבב אותיות ומספרים:

```
PRINT 2 ,"ABC"
```

שים לב שיש מרווח בין 2 ל־ABC על המסך. עכשיו הקש:

```
PRINT 2;"ABC"
```

ואז הקש `PRINT 2 "ABC"`

שימוש בפסיק בין הפריטים לאחר הפקודה PRINT מרווח 16 עמודות. שימוש בנקודה פסיק אינו משאיר רווח ואי שימוש בסימן כלשהוא גורם לשגיאה ניתן להשתמש ב־PRINT עם כל הסימנים המתמטיים שעל לוח המקשים כך שהמחשב הופך למחשב כיס. לדוגמה:

```
PRINT 2+2
```

התשובה תופיע בחלק העליון של המסך. השוה זאת עם:

```
PRINT "2+2"
```

ניתן לשלב זאת על מנת לקבל תוצאה שימושית יותר
נסה:

```
PRINT "2+2  =";2+2
```

נסה פעולות חשבון אחרות כמו:

```
PRINT 3—2
PRINT 4/5
PRINT 12*2
```

בסימן * משתמשים למטרת כפל במקום בסימן X, על מנת להמנע מבלבול באות X האנגלית. בסימן / כסימן החלוקה (ולא בסימן ÷).

נסה חישובים שונים. אם תרצה תוכל להשתמש במספרים שליליים וגם במספרים בעלי נקודה עשרונית. אם תגיע למצב שבו השתמשת ב־22 שורות של החלק העליון במסך תגלה שיקרה משהוא מעניין. כל הכתוב יעלה למעלה בשורה אחת והשורה העליונה תעלם. לתופעה זו קוראים scroling. החישובים לא תמיד בסדר שאתה עשוי לצפות. לדוגמה נסה:

```
PRINT 2+3*5
```

אתה עשוי לצפות שהמחשב יחבר 2 עם שלוש ויקבל 5 ואז יכפיל ב־5 ויקבל 25. כפל וחילוק מבוצעים תמיד לפני חיבור וחיסור כך שהביטוי '2+3*5' משמעותו קח את 3 הכפל ב־5 (נותן 15) ואז הוסף 2 הנותן 17. זו תהיה התוצאה שתופיע על המסך.

מאחר שכפל וחילוק מבוצעים קודם נאמר שיש להם עדיפות גבוהה יותר מאשר חיבור וחיסור. לכפל וחילוק ישנה אותה עדיפות והם יבוצעו על פי סדר הכתיבה משמאל לימין.
לאחר שהמחשב ביצע את כל פעולות החילוק והכפל נשארו פעולות החיבור והחיסור וגם בין אלה ישנה אותה עדיפות והן יבוצעו משמאל לימין.
נראה איך המחשב יבצע את

```
PRINT 2Ø—2*9 + 4/2*3
```

בתחילה מבוצעת פעולת הכפל והחלוק על פי הסדר משמאל לימין ואחר כך החיבורים והחיסורים

```
I 2Ø—2*9 + 4/2*3
II 2Ø — 18+4/2*3
III 2Ø — 18+2*3
IV 2Ø — 18+6
V 2+6
VI 8
```

אפילו שאנו צריכים לדעת אם לפעולה יש עדיפות גבוהה או נמוכה המחשב עושה זאת ע"י התאמת מספרים בין 1 ל־16 שייצגו את העדיפות עבור כל פעולה: * ו / עדיפות 8 ל+ יש עדיפות 6. סדר פעולות זה נוקשה באופן מוחלט אבל תוכל לעקפו ע"י שימוש בסוגריים. כל דבר בתוך סוגריים יחושב ראשון ולאחר מכן יטופל כמספר אחד, כך ש:

```
PRINT 3*2 + 2
```

יתן את התשובה 8 = 2 + 6
אבל:

```
PRINT 3*(2 + 2)
```

יתן את התשובה 12 = 3*4.

יהיה זה שימושי לפעמים לתת למחשב בטויים כאלה (2+2), כאשר המחשב מצפה ממך למספר תוכל לתת לו ביטוי מתמטי במקום מספר, והוא כבר יחשב את התשובה. היוצאים מן הכלל לכל זה הם כל כך מועטים שהם יודגשו באופן ברור לכל מקרה ומקרה.

תוכל לכתוב מספרים בעלי נקודה עשרונית (ע"י שימוש בנקודה.) כמובן תוכל להשתמש בייצוג מעריכי המקובל למדי במחשבי הכיס המדעיים.

במקרה זה לאחר מספר רגיל (עם או בלי נקודה עשרונית) תכתוב את החלק המעריכי (exponent) כשהוא מכיל את האות e ואז אולי (מינוס) ואת המספר. החלק המעריכי יזיז את הנקודה העשרונית ימינה (או שמאלה במעריך שלילי), וכך יוכפלו או יחולקו הספרות האורגינליות ב־1Ø כמה פעמים לדוגמא:

```
2.34 eØ = 2.34
2.34 e3 = 234Ø
2.34 e—2 = Ø.Ø234
```

וכך הלאה.
נסה להדפיס ביטויים אלה על המחשב. במקרים בודדים לא ניתן להחליף מספר בביטוי מתמטי. לדוגמא תוכל לכתוב:

```
(1.34+1) e (6/2)
```

ישנם גם ביטויים שתוצאתם אינה ערך אלא מחרוזת של אותיות. ראית צורות פשוטות אלה פעמים רבות כמחרוזת האותיות הסגורות בתוך גרשיים מצידיהן.

תופעה זו מקבילה לצורות הפשוטות של ביטויים ספרתיים אשר הינם בעצם רק מספר בפני עצמו. מה שלא ראית עד כה הוא השימוש של ה + במחרוזות (אבל לעולם לא —,* או / וזאת על מנת שלא יהיו בעיות בסדר העדיפויות) חיבור מחרוזות פשוט מצמידם זו לזו בשרשרת האחת לאחר השניה נסה:

```
PRINT "jers" + "ey cow"
```

תוכל לחבר במה מחרוזות שתרצה בשורה אחת יתקבל ביטוי אחד מאורך. אם תרצה תוכל להשתמש גם בסוגריים.

פקודות פשוטות

אפשר להשתמש בזכרון המחשב על מנת לאחסן דברים שונים. עד כה ראינו כי הפקודה PRINT מאפשרת להדפיס על הצג אותיות, מספרים ותוצאות של חישובים, כאשר האותיות והמספרים יכולים להופיע באופן מעורב.

אם נרצה לפקוד על המחשב שיזכור מספר או מחרוזת של אותות נצטרך להקצות חלק מהזכרון לשימוש הזה.

לרוב המחשבים ישנו מקש הנקרא memory (זכרון) במקש זה משתמשים כדי לזכור מספר או אות המחשב שלך יכול לעשות הרבה יותר מזה, הוא יכול להשתמש בהרבה תאי זכרון דמיוניים כרצונך ותוכל לתת שם לכל אחד מהם.

לדוגמא: נניח שברצונך לזכור את גילך. כדוגמה — 34 משתמשים בפקודה LET (LET היא מלת מפתח המודפסת על המקש L וכותבים:

LET age = 34

מה שיקרה כאשר משתמשים בפקודה LET הוא שחלק מסוים מהזכרון יסומן ב־age (גיל) והמספר 34 יאוחסן בתוכו. על מנת לקבל את המידע המאוכסן הדפס:

PRINT age

ואז יופיע המספר 34. קל מאוד לשנות את התוכן של תא הזכרון שקראנו לו age הקש:

LET age = 56

והדפס:

PRINT age

56 יופיע על המסך age היא דוגמה למשתנה שניתן לשנות את ערכיו. אפשרי לשלב הדפסת הודעה ישירות למסך עם ערך של משתנה לדוגמה הדפס:

PRINT "your age is" ;age

על כל פנים, המחשב הוא הרבה יותר שימושי מזכירת מספרים ששמות מחוברים אליהם. הוא יכול גם לזכור מחרוזות של אותיות. כדי להבחין בין משתנים מספריים למשתנים מחרוזתיים משתמשים בסימן הדולר $ בסוף שם המשתנה. לדומה אם נרצה שישמור את המחרוזת "your age is" נוכל לקרוא לה a$ (שמות של משתנים מחרוזתיים יוכלו להיות בעלי אות אחת בלבד כאשר יש בהם את ה־$) לכן הדפס:

LET a $ = "your ageis"

אם עכשיו תדפיס "PRINT a$"

תופיע על הצג כל מחרוזת האותיות.

אם לא כיבית את המחשב מתחילת הפרק הדפס

PRINT a$;age

ותראה מה יקרה

ישנן דרכים אחרות על מנת להכניס מידע לזכרון המחשב מבלי להשתמש בפקודה LET לדוגמה הפקודה INPUT בצורתה הפשוטה אומרת למחשב כי מידע כלשהוא יגיע אליו מן המקשים. בכל פעם במקום להדפיס LET aeg תוכל להדפיס

PRINT age

כאשר תלחץ על מקש ה־ENTER מצביע מהבהב מסוג L יופיע על הצג. זה אומר שהמחשב מחכה למידע ממך ולכן הדפס את מגילך ואז לחץ על ה־ENTER. אף על פי שנראה לך ששום דבר אינו קורה למשתנה ניתן עכשיו הערך שהדפסת.

הדפסת PRINT age

תוכיח זאת.

הבא נשלב כל זאת ביחד למערכת של פקודות.

הדפס:

```
LET b$ = "What is yovr age?"
LET a$ = "Your age is"
INPUT (b$) ;age :PRINT a$ ;age
```

שים לב שהשורה האחרונה מכילה שתי פקודות מופרדות ע"י הנקודותיים

INPUT (b$) ;age

הינה דרך אחרת להכניס

INPUT "What is your age?" ;age

תכנות בסיסי

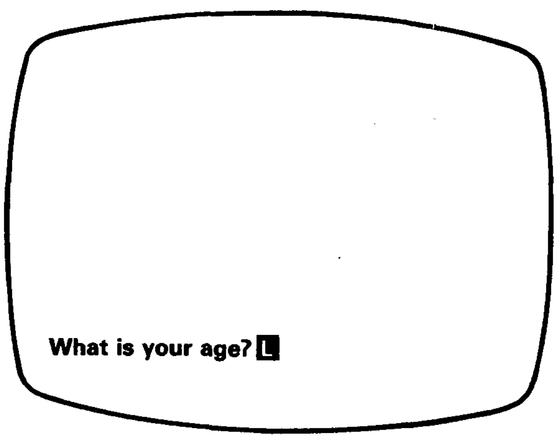
עד כה פקדנו על המחשב ישירות מלוח המקשים. אמנם אפשר לשלב מספר פקודות ביחד אך השימוש בשיטה זו מוגבל למדי.
עוצמת המחשבים היא היכולת לתכנתם, המשמעות היא שנוכל לתת להם סדרת הוראות לביצוע דברים כסדרם. לכל מחשב ישנה שפה משלו המאפשרת לנו ל״דבר״ איתו. ישנן שפות פשוטות שהמחשב יכול להבינן, לרוע המזל השפות הפשוטות להבנה ע״י המחשב קשות להבנה לבני אדם גם ההפך נבון: שפות הקלות להבנה לבני אדם הינן מסובכות יחסית עבור המחשב, ולפעמים אפילו יש לתרגמן או לפרשן עבורו.
ה־ZX SPECTRUM משתמש בשפה עילית הנקראת BASIC ראשי התיבות של המלה BASIC הם:

Beginners All-purpose Instruction Code Symbolic

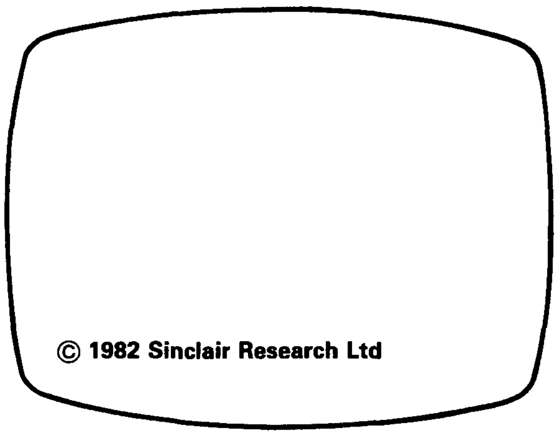
שפרושו: צופן הוראות למתחילים, בסמלים ולכל מטרה. השפה עוצבה במכללת דרטמות בניו המפשר ארה״ב ב־1964. שפה זו נפוצה מאוד במחשבים אישיים. אפע״פ שהשפה די דומה בין המחשבים האישיים, ישנם הבדלים משמעותיים. זאת הסיבה שהוראות אלו נכתבו במיוחד עבור ה־Zx spectrurn, ה־BASIC של הספקטרום אינו שונה מאוד מה־BASIC המקובל ולא תתקל בבעיות רבות להבינו. ה־BASIC של הספקטרום (שלא כבמחשבים אחרים) אינו מרשה להשמיט את הפקודה LET כאשר מייחסים ערכים ומשתנים, ישנו גבול למספר ההוראות שאפשר לאחסן במחשב. ה־ZX Spectrum יודיע לך שהגעת לגבול זה ע״י זמזום.
כאשר מתכנתים ב־BASIC הכרחי להביא לידיעת המחשב את סדר ההוראות אותם עליו לבצע ולכן כל שורה בסדר זה היא בעלת מיספר בתחילתה. מקובל להתחיל ב־10 ולהוסף 10 בכל שורה כך שאפשר יהיה להשחיל שורות נוספות במקרים שהיו השמטות או שדרושים שינויים בתכנית. נסתכל בתוכנית פשוטה, נתייחס לסדרת הפקודות שבסוף הפרק הקודם, אם נרצה לחזור על סדרה זו צריך יהיה להכניס כל פעם. התוכנית מתגברת על בעיה זו. הדפס את השורות דלהלן עם ENTER בגמר בל שורה:

```
1Ø LET b$ = "what is your age?"
2Ø LET a$ = "your age is"
3Ø INPUT (b$) ;age
4Ø PRINT a$ ;age
```

שים לב שאין צורך להכניס רווחים, מלבד בתוך הגרשיים. דבר לא יקרה עד שנפקוד על המחשב להתחיל לפעול על התוכנית, ע״י שימוש ב־RUN (זו מלת מפתח על המקש R הכנס פקודה זו וראה מה יקרה.



שים לב לחץ המראה ימינה על כל שורה שהוכנסה בתכנית.חץ זה מראה את השורה האחרונה שהכנסת. אם תרצה לראות את כל התכנית מחדש הקש ENTER (או LIST).
תוכל להשתמש ב־RUN על מנת להריץ את התוכנית כמה פעמים כרצונך. כאשר לא תצטרך יותר בתוכנית זו תוכל לסלקה ע״י שימוש בפקודה NEW. פקודה זו מוחקת את התוכנית המאוכסנת בזכרון ומשאירה לוח חלק המוכן להכנסת תוכנית חדשה.
הקש NEW ואחר כך LIST וראה מה קורה.



בסיכום:
כאשר אתה מדפיס פקודה שלפניה מספר, המחשב יקבל פקודה זאת כשורה בתוכנית ולא כפקודה סתמית. המחשב לא יבצע אותה מיד אלא יאחסן שורה זו לשימוש בהמשך.
לעזרתך ה־Zx spectrum כותב על הצג את כל שורות התוכנית שהכנסת וה־< מצביע על השורה האחרונה שהוכנסה.
המחשב לא יבצע שורות אלה מיד אלא יאחסנן בזכרונו. על מנת שהמחשב יבצע שורות אלה, השתמש בפקודה RUN. אם תלחץ על ENTER תקבל את שורות התוכנית בחזרה.
נבחן תוכנית פשוטה אחרת, תוכנית זו תהיה יותר מתמטית היא תדפיס את כל ריבועי המספרים בין 1 ל־1Ø. (ריבוע של מספר מוגדר כמספר המוכפל בעצמו).
על מנת לקרוא למספרים מ־1 עד 1Ø נשתמש בתפיסת תכנות נוספת שיש ל־BASIC. זאת היא שיטה שבעזרתה נוכל להביא את המחשב לספור מוקדם יותר. ראינו שניתן לאחסן מספרים בזכרון המחשב ע״י הצמדת שם אליהם. או בשפה טכנית ע״י הצמדת ערך למשתנה. אם נרצה שהמשתנה X יתחיל בערך 1 ויתקדם בצעדים של 1 עד ל־1Ø נעשה זאת ע״י שימוש בפקודה FOR ...TO ...STEP.
על מנת להכניס תוכנית זו עליך להקיש NEW כדי להפטר מכל מה שהיה קודם במחשב כעת הדפס:

```
1Ø FOR X = 1 TO 1Ø STEP 1
```

(בד״כ את ה־1 STEP אפשר להשמיט אם הספירה נעשית בצעדים של 1) השורה הבאה צריכה לומר למחשב מה לבצע על X ללא קשר לערך שהוא מיצג. הדפס:

```
2Ø PRINT X, X*X
```

המחשב זקוק לשורה שתאמר לו לנוע לערך הבא של X ולכן הדפס:

```
3Ø NEXT X
```

כאשר המחשב יגיע להוראה זו הוא יעבור לשורה 10, ואז יחזור על התהליך. כאשר X יעבור את הערך 1Ø המחשב ילך לשורה הבאה בתכנית כלומר שורה 40.

עכשיו התוכנית צריכה להופיע על המסך בצורה הבאה:

```
1Ø FOR X = 1 TO 10 STEP 1
2Ø PRINT X,X*X
3Ø NEXT X
```

להשלמה דרושה שורה נוספת שתאמר למחשב שהתכנית נגמרת כאשר X=10 לכן הדפס:

```
4Ø STOP
```

לאחר הפקודה RUN יופיעו שני טורים. טור אחד עם ערכי X והשני עם הערך של X*X (או X^2) ניתן לתת כותרת לעמודות אלה ע"י הוספת השורה:

```
5 PRINT "X" ,"X*X"
```

שים לב שאע"פ שהכנסנו שורה זו אחרי כל השורות המחשב מכניס שורה זו במקומה הנכון באופן אוטומטי לפי מספרה. נסה לכתוב תוכניות אחרות ע"י כתיבת פונקציות מתמטיות אחרות אם יש לך ספק כלשהוא בצורת השימוש בהן פנה לעמודים המתאימים בספר הוראות לתכנית ב־BASIC.

שימוש ברשם קול

כדי להמנע מהדפסת תוכניות מחדש למחשב בכל פעם ה־Zx Spectrum יכול להקליט תוכנית על קסטות בעזרת רשם הקול הביתי הרגיל, אם יש לך תוכנית בזכרון נסה לשמור אותה תוך שימוש בהוראות הבאות.

אם ניתן לשמור תוכניות על קסטות ניתן גם להעלותם בחזרה מאוחר יותר אל תוך המחשב. רוב רשמי הקול יתאימו, מבחינת המחשב רשם קול רגיל המקליט בערוץ אחד (מונו) היא לפחות טוב כמו רשם קול יקר סטראופוני. ועושה פחות בעיות. תמצא שהמיספור ברשם הקול הוא שימושי מאוד.

לרשם הקול חייבת להיות כניסה למיקרופון ויציאה לאוזניות (אם אין, נסה את היציאה החיצונית לרמקול). הם צריכים להיות תקעי 3.5 מ"מ (כדי להתאים לחוטים המוליכים שסופקו לך) בגלל שסוגים אחרים עלולים לא לתת אותות מספיק חזקים בשביל המחשב. כל קסטה רגילה תתאים אע"פ שקסטות של low noise יוכלו להיות טובות יותר. לאחר שהשגת רשם קול מתאים חבר אותו למחשב ע"י המוליכים שקבלת עם ה־Zx spectrum. חוט מוליך אחד צריך לחבר את שקע הכניסה למיקרופון שבטייפ לשקע המסומן ב־'MIC' שבצד האחורי של המחשב. החוט השני צריך לחבר את שקע היציאה לאוזניות ברשם הקול לשקע 'EAR' שבמחשב (לא תוכל לגרום נזק ל- spectrum ע"י חיבור לא נכון של החוטים).

כאשר תשתמש בפקודת ה־SAVE כדי לשמור תוכנית על קסטה ודא שאחד התקעים המחברים את יציאת האוזניה בטייפ או כניסת ה־EAR במחשב מנותקים כדי לא לגרום ללולאה שתגרום להקלטה שגויה של התכנית. הדפס תוכנית כלשהיא אל תוך המחשב, נניח תוכנית הריבועים שבפרק הקודם, ואז הדפס

```
SAVE "Squares"
```

Squares הוא פשוט שם שניתן להשתמש בו כדי לסמן את התוכנית בזמן שהיא על הקסטה. תוכל לתת שמות בעלי 10 סימנים לכל היותר.

המחשב יגיב בהודעה Start tape then press any key בתחילה נסביר מה צריך לקרות לפני שתקליט את התוכנית. אל תלחץ על הטייפ אלא על מקש ב־Zx Spectrum והסתכל בגבול של צג הטלויזיה. תראה פסים אופקיים צבעוניים.

5 שניות של פסים אדומים וכחולים בהירים בערך בעובי 1 ס"מ שזזים באטיות כלפי מעלה, אז התפרצות מהירה של פסים צהוכים וכחולים, שניה אחת הכל כרגיל ו־2 שניות של אדום וכחול בהיר שוב ואז בערך שניה אחת של הפסים הצהובים והכחולים שוב. נסה זאת שוב עד שתזהה ותכיר כל סוג של צבעים. המידע נשמר בשני חלקים ולפני כל חלק ניתן לראות את הפסים האדומים הכחולים בהירים. המידע עצמו מועלה בו זמנית עם הופעת הפסים הצהובים כחולים. החלק הראשון מכיל את שם התוכנית ומידע כללי נוסף ואילו השני מכיל את התוכנית עצמה ביחד עם כל המשתנים הקיימים הקטע הלבן (הרגיל) שבינהם הוא רק מירווח. כעת נקליט את התוכנית על קסטה.

1) הזז את סרט הקסטה למקום ריק או למקום שאתה מוכן למחוק.

2) הדפס:

```
SAVE "squares" (ENTER)
```

3) התחל להקליט בטייפ (ברוב רשמי הקול יש ללחוץ על play וְ record בו זמנית).

4) לחץ על כפתור כלשהוא ב־Zx spectrum.

5) התבונן בטלויזיה כמקודם וכאשר המחשב גמר להעביר את התכנית לטייפ תופיע ההודעה OK Ø ואז עצור את רשם הקול.

על מנת לודא שהצלחת תוכל לבדוק את האותות שעל הקסטה כנגד התוכנית שבמחשב. הבדיקה תעשה ע״י שימוש בפקודה VERIFY.

1) כוון את כפתור העוצמה (Volume) ברשם הקול לבערך מחצית הדרך. וחבר את החוט של האוזניות EAR.

2) גלגל בחזרה את הקסטה למקום כלשהוא לפני ההקלטה.

3) הדפס:

VERIFY "Squares"

(VERIFY נמצא במצב ההרחבה E ואז shiftְR).

4) הפעל את רשם הקול לחץ על (play)

גבול הטלויזיה ישנה צבעיו אדום ואו כחול בהיר עד שהטייפ יגיע אל התוכנית המוקלטת — אז תראה את אותם הפסים שראית בזמן שמירת התוכנית (ע״י SAVE במרווח הזמן שבין הפסים תרשם על המסך ההודעה Program Squares. כאשר המחשב מחפש מידע על הטייפ הוא מדפיס את השם של כל דבר שבו יפגוש. אם תראה את כל הפסים ואחרי כן המחשב יעצור, עם דיווח ∅O K דע שתוכניתך הוקלטה ונשמרה על הקסטה, במידה ומשהוא השתבש עליך לשאול את השאלות הבאות על מנת למצוא מה השתבש.

איך תוודא שתוכניתך נשמרה?

אם לא הופיע שם התוכנית יכולות להיות שתי תקלות. או שהתכנית לא נשמרה כראוי, או שהיא לא נקראת בחזרה כראוי. בכל מקרה עליך למצוא את סוג התקלה. על מנת לברר אם התכנית נשמרה כראוי גלגל את הקסטה עד למקום שלפני התחלת ההקלטה ונגן אותה דרך הרמקול של רשם הקול (קרוב לודא י שתִצטרך לנתק את החוט משקע האוזניות של הטייפ) הפסים האדומים והכחולים מייצרים צליל גבוה ברור ויציב והמידע הכחול והצהוב משמיע קולות הרבה פחות נעימים בדומה להודעה במורס. שני סוגי הצלילים חזקים למדי כאשר מפעילים עוצמה מלאה, ויכולים בקלות להתגבר על קול שיחה. במידה ולא תשמע רעשים אלה תוכניתך קרוב לודאי שלא נשמרה.

בדוק אם החוטים מחוברים לשקעים המתאימים, ודא ששקעי המיקרופון (MIC) מחוברים. ודא ששקעי האוזניה (EAR) אינם מחוברים. קורא שבסוגים מסוימים של רשמי קול, התקעים אינם יוצרים מגע כאשר הם נדחפים ישר פנימה. נסה למשוך אותם החוצה בערך 2.5 מ״מ ואז תוכל לפעמים להרגישם מתיצבים בעֶמדת נעילה.

בדוק שלא בצעת הקלטה על הקטע מהפלסטיק בתחילת סרט הקסטה. לאחר שבדקת זאת נסה לשמור את התוכנית מחדש.

אם תוכל לשמוע את הקולות שתוארו מקודם אז פעולת ה־SAVE התבצעה קרוב לודאי כראוי והבעיה היא בקריאת התוכנית אל תוך המחשב.

בדוק שוב את החוטים, בדוק את עוצמת הקול (Volume) אם העוצמה נמוכה מדי המחשב לא ישמע את האותות היטב ולכן לא תוכל לראות את הפסים הנכונים על המסך, אך אם העוצמה חזקה מדי האותות יגיעו בעוותים, ואז תוכל לשמוע אותם דרך הרמקול הפנימי של המחשב. ישנו תחום רחב של ערכי ביניים בין שני המצבים. ותוכל למצוא אותו ע״י מספר ניסוים.

מקרה נוסף הינו כאשר המחשב מזהה את התוכנית, מדפיס את שמה אבל עדיין משהוא משתבש. האפשרויות הן שהדפסת לא נכון את השם בשלב ה־SAVE (אז המחשב יכתוב את השם השגוי על גבי הצג) או ששגית בזמן ה־VERIFY ואז המחשב יתעלם מהתוכנית וימשיך להבזיק אדום וכחול בגבול המסך.

כאשר ישנה שגיאת אימות עם תוכן הקסטה המחשב יחזור עם ההודעה

R Tape loading error

שפרושה במקרה זה שהמחשב אינו מצליח לאמת את התוכנית. במקרה זה בצע שמירת תוכנית מחודשת (SAVE).

יש אפשרות שכיוון העוצמה (Volume) ברשם הקול אינו לגמרי בסדר אבל אינו רחוק מהרצוי מאחר שהמחשב הצליח לקרוא את החלק הראשון

הבה נניח שהצלחת לשמור את התוכנית וכן לאמתה (ע״י Verify) הטענתה מחדש מתבצעת בדיוק כמו האימות מלבד צורת הפקודה שהיא:

LOAD "Squres"

VERIFY "Squares"

LOAD נמצא על המקש J. אם הצלחת לבצע אימות נכון לא תהיה שום בעיה בטעינה. פקודת הטעינה (LOAD) מוחקת את התוכנית הישנה ומשתניה במחשב לפני פעולת ההטענה עצמה. כאשר תוכנית הוטענה ניתן להריצה ע״י הפקודה RUN. ניתן לקנות תוכניות מוקלטות מראש על קסטות. עליהן להיות מוכנות במיוחד ל־Zx spectrum מאחר ודרכי ההקלטה שונות ממחשב למחשב. כדי להבחין בין התוכניות המוקלטות על אותו צד בקסטה לכל תוכנית יש שם. תוכל לבחור את התוכנית שתרצה ע״י פקודת ההטענה והדפסת השם. לדוגמה תוכנית בשם helicopter תועלה למחשב ע״י הפקודה:

LOAD "helicopter"

(הפקודה "" LOAD משמעותה הטען את התוכנית הראשונה בה תתקל. פקודה זו יעילה ביותר אם שכחת את שם התוכנית).

צבעים

אחת הסיבות לקנית Zx spectrum היא היכולת להשתמש בצבעים על מסך הטלויזיה. המסך מחולק לשני אזורים אזור חיצוני — BORDER גבול) ואזור מרכזי — PAPER (נייר). ניתן לשנות את הצבעים של־שני אזורים אלה כרצונך. וזאת ע״י שימוש בלוח המקשים או ע״י תכנית. אפשר להשתמש ב־Zx Spectrum בשמונה צבעים. הצבעים מוספרו מ Ø עד 7. אע״פ שהצבעים נראים כמסודרים בסדר מקרי הם נותנים דרגות הולכות ופוחתות של צבע אפור על גבי מסך טלויזית השחור לבן. להלן רשימת הצבעים ומספריהם הרשומים גם מעל מקשי המספרים:

Ø **שחור** — black
1 **כחול** — blue
2 **אדום** — red
3 **סגול** — magenta (purple)
4 **ירוק** — green
5 **תכלת** — cyan (pale blue)
6 **צהוב** — yellow
7 **לבן** — white

כאשר המחשב מופעל פועלת המערכת כתחילה בשחור לבן. כך שהערך הנורמלי של ה־BORDER (גבול) ושל ה־PAPER (נייר) הוא 7 דהיינו לבן. צבע כל אות מספר וכו׳ שיופיעו על המסך יוגדר ע״י פקודת INK שערכה הראשוני Ø דהיינו שחור. שלוש הפקודות השולטות על צבעי המסך נקבעות ע״י המחשב. אך ניתן לשנות ערכים אלה לדוגמה ע״י הקשת.

BORDER 2

אם תזכור להקיש ENTER הגבול ישתנה מלבן לאדום כולל האזור התחתון של המסך בו ההוראות והפקודות מודפסות. נסה לקבוע ערכים אחרים ותראה את שינוי הצבע. נסה לשנות את מרכז המסך ע״י הקשת:

PAPER 5

פקודה זו נכתבת במצב ההרחבה E שתואר מוקדם יותר, הוא מושג ע״י הקשה על CAPS SHIFT וגם SYMBOL SHIFT באותו זמן. ה־PAPER עצמו מושג ע״י SHIFT ואז C וכאשר ה־ENTER ילחץ פעמיים החלק המרכזי יקבל צבע תכלת הקשת ה־ENTER הראשונה מכטלת את פקודת ה־ENTER שכבר אוחסנה כמחשב, אולם רק כאשר יוקש ה־PAPER החדש לשימוש (הלחיצה השניה גורמת למחשב לבצע LIST לכל תוכנית ולכן בונה מחדש את נתוני המסך) כאשר אתה משתמש בטלויזיה צבעונית והצבעים לא השתנו, נסה להתאים את בקרת הצבע שבטלויזיה ומעט את בקרות כיוון התחנה. פקודת ה־INK זהה לפקודת ה־PAPER ומווסתת את הצבעים של האותיות והמספרים המופיעים באזור ה־PAPER במסך. מובן מאליו שאם צבעי ה־INK וה־PAPER זהים לא תוכל לקרוא את המידע על המסך.

בפקודות BORDER, PAPER וְINK ניתן להשתמש כתוך תוכניות לדוגמה תוכנית פשוטה שתראה את טווח הצבעים האפשרי

```
10 FOR x=0 TO 7
20 BORDER x
30 PAPER 7-x: CLS
40 PAUSE 50
50 NEXT x
```


כאשר תוכנית זו מורצת (RUN) היא מראה את שמונת הצבעים תוך יצירת ניגוד **בין** צבעי ה־PAPER **ל־**BORDER. פקודת ה־CLS (נקה מסך) **לאחר** ה־PAPER מכריחה את המחשב לבנות מחדש את **צורת** המסך תוך שימוש **בצבע** ה־PAPER החדש. פקודת ה־PAUSE (השהיה) עוצרת את המחשב **למשך** שניה אחת כדי שנוכל לראות מה קורה (נסה להריץ את התוכנית בלי שורת ה־PAUSE). **כדי** לראות כיצד פקודת ה־INK פועלת הדפס אתַ התוכנית הבאהִ לאחר פקודת NEW.

```
1Ø BORDER 7
2Ø PAPER 1
3Ø INK 4
4Ø PRINT "Green characters on blue background"
```

התוכנית תדפיס אותיות ירוקות על רקע **כחול**.

ישנן פקודות נוספות שקשורות ליכולתו הצבעונית של ה־Zx spectrum המפורטות במדריך התכנות **ב־**BASIC.

קולות

ה־Zx spectrum יכול ליצור קולות במגוון אינסופי. תדירות הקול ואורכו ניתנים לשליטה ע״י המשתמש.

בפקודה BEEP משתמשים על מנת לפקוד על המחשב להוציא צליל. BEEP נמצא במצב המורחב E ומושג ע״י שימוש במקש ה־Z.

הצליל המרכזי בפקודת BEEP הוא דו אמצעי (middle C) הוא יכול להשתנות בתוך פקודת ה־BEEP ואפשר להשיג כל תו אם הוא מבוטא כצליל או תת צליל מעל או מתחת לדו (C), כאשר תתקבל הפקודה:

BEEP 2 ,Ø

המחשב יוציא את הצליל דו למשך 2 שניות.

המספר הראשון לאחר הפקודה יתן את אורך הצליל בשניות ואילו המספר השני יקבע את גובה התו מעל או מתחת לדו זה. מכאן שהערך שיתן את הצליל דו הוא Ø והערך שיתן דו דיאז (C #) הוא 1. הערך עבור רֶה (D) הוא 2 וכן הלאה עד הדו (C) הבא שערכו 12 מפני ש־12 דרגות של צלילים וחלקיהם יתנו אוקטבה שלמה. תוכל להמשיך עד 13 ומעבר לזה אם תרצה וככל שהמספר יהיה יותר גבוה כך יהיה גובה הצליל שיופק ע״י המחשב. נסה:

BEEP 1 ,4 :BEEP 1,2 :BEEP 2 ,Ø

תוכל לשמוע את התיבה הראשונה של השיר ״שלושה עכברים עִוורים״. מאחר שניתן לצרף כמה פקודות BEEP יחדיו בעזרת הנקודותיים, תוכל לייצר, אם יש לך את הסבלנות, שיר שלם. אפשר להשתמש בנקודותיים לא רק על מנת לחבר מספר פקודות BEEP. אפשר גם להשתמש בהן על מנת לבנות פקודות המורכבות מכל הפקודות היסודיות. כדוגמא מסובכת יותר תוכל ליצור פקודה של קולות וצבעים משתנים ע״י עכוב של BEEP וגם BORDER יחדיו:

BORDER 1 :BEEP 1 ,14 :BORDER 3 :BEEP 1 ,16 :BORDER 4:
BEEP 1 ,12 :BORDER 6 :BEEP 1 ,0 :BORDER 5 :BEEP :BORDER 1

אל תחשוש מהעובדה שפקודה זו מתמשכת יותר מאשר שורה אחת המחשב לא ישים לב לעובדה זו.

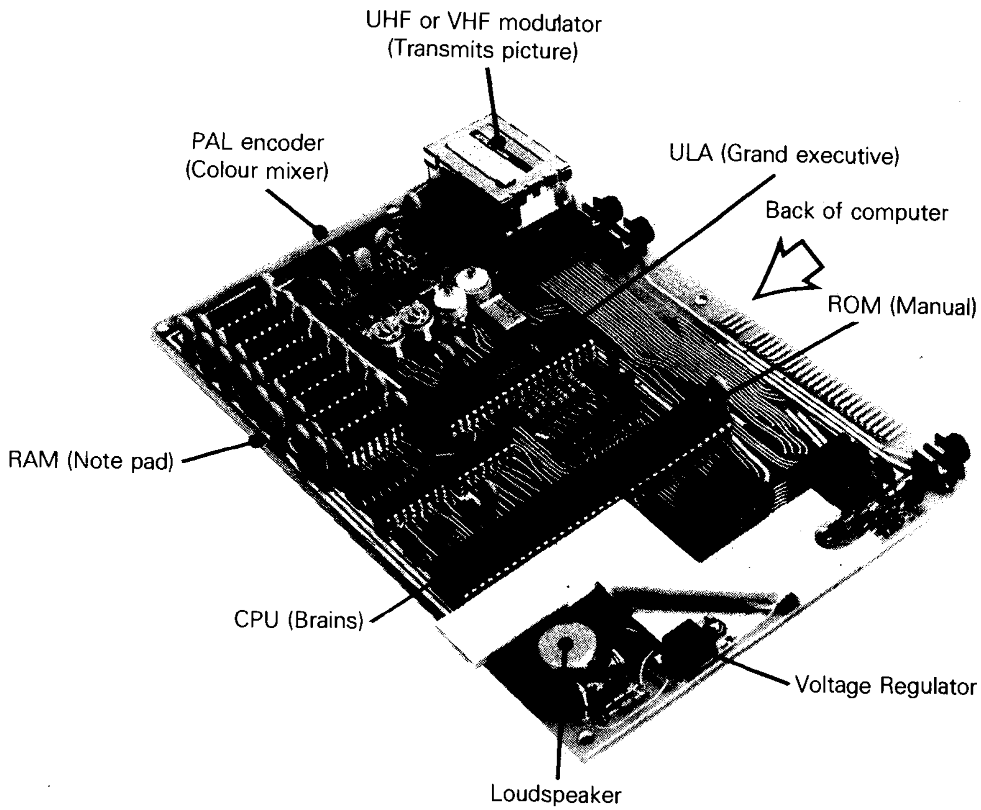
הנה תכנית קצרה על מנת לנגן מערכת שלמה של צלילים:

```
1Ø FOR X = Ø TO 24
2Ø BEEP 2 ,X
3Ø NEXT X
```

יש להדגיש כי תוים נמוכים מדו (C) משתמשים במספרים שליליים, תוכל לבצע דברים רבים נוספים אם תסתכל במדריך התכנות ב־BASIC.

פנים המחשב

התמונה כאן מראה את תוכו של ה־Zx Spectrum.



כפי שתראה כל חלק מתואר בשלוש אותיות בראשי תיבות. חתיכות הפלסטיק המלבניות שחורות בעלות הרגליים הרבות הינן המעגלים המשולבים (שבבי הסיליקון) אשר למעשה מבצעים את כל העבודה. בכל אחד ישנו ריבוע שצלעו 6.3 מ״מ של סיליקון המחובר בחוטי מתכת לרגלים המתכתיות. על שבב הסיליקון הזה ישנם אלפי טרנסיסטורים היוצרים את המעגלים האלקטרוניים שהינם המחשב. ה״מוח״ של המחשב הינו שבב עיבוד הנתונים הנקרא לעיתים קרובות CPU (Central processor unit — יחידת עיבוד נתונים מרכזית).

ה־CPU המסוים הזה נקרא Z80A שהיא גירסה מהירה **יותר של ה־Z80** הפופולרי.

מעבד הנתונים (CPU) שולט במחשב, מבצע את החישובים, בודק איזה מקש נלחץ, מחליט מה לעשות בתגובה ובכלל מחליט מה המחשב צריך לעשות. על אף כל חוכמתו לא יכול המחשב לבצע הכל בעצמו. למשל הוא אינו יודע דבר על BASIC או חשבון עשרוני. בעניינים אלה עליו לקבל את ההוראות משבב אחר. הוא שבב ה־ROM (READ ONLY MEMORY — זכרון קריא בלבד) ה־ROM מכיל רשימה ארוכה של הוראות שהן תוכנית המחשב. הן אומרות למעבד הנתונים מה לעשות בכל הנסיבות הצפויות מראש. תכנית זו לא נכתבה ב־BASIC אלא במה שנקרא שפת המכונה של Z80. צורת התוכנית הינה רשימה ארוכה של מספרים. ביחד ישנם 16,384 מספרים כאלה ולכן נקרא ה־BASIC של ה־BASIC Zx spectrum של 16K בROM. 1K הוא 1024. על אף שישנם מספר שבבים זהים למחשב זה במחשבים אחרים, שרשרת ההוראות הזאת הינה בלעדית ל־Zx spectrum ונכתבה במיוחד עבורו.

שמונה השבבים שלה: ה־CPU הינם הזכרון. זהוא ה־RAM (Random Aecess Memory — זכרון בעל גישה חופשית לכל תא מקרי שניתן גם לכתוב וגם לקרוא ממנו) ה־RAM הוא המקום בו ה־CPU מאחסן מידע שהוא רוצה לשמור. בזכרון נמצאות כל תכניות ה־BASIC, המשתנים, תמונת מסך הטלויזיה ומספר פריטים אחרים אשר משמשים למעקב אחר מצב המחשב.

ישנם שני שבבים אחרים הפועלים בשיתוף עם ה־RAM ובקרבתו. השבב הגדול נקראים SCL (Logic Sinclair computer) שפועל כמרכז התקשורת ומודא שכל דרישות המעבד יבוצעו.

כמו כן הוא קורא את הזכרון על מנת לראות מה מכילה תמונת הטלויזיה ושולח את האותות המתאימים אל הטלויזיה.

משדר ה־PAL הוא קבוצה שלמה של אביזרים הממירים את האותיות עבור הטלויזיה הצבעונית. המייצב (רגולטור) ממיר את המתח הבלתי יציב לספק הכוח ל־5 וולטים יציבים לחלוטין.

תם פרק המבוא

אם אתה חש שהבנת אותו היטב אנו ממליצים שתנסה כעת לקרוא את המדריך לתכנות ב־BASIC.

פרק 1: המדריך לתכנות ב־BASIC

באם קראת תחילה את פרק המבוא, או התחלת ישירות עם ספר זה, הנך אמור לדעת שפקודות מתבצעות מיידית וכי הוראות מתחילות עם מספר השורה כשהן נאגרות כדי לפעול מאוחר יותר. הנך אמור להכיר את הפקודות LET, PRINT וכן INPUT (אשר ניתנות לשימוש בכל המערכות המשתמשות בשפת — BASIC) ואת הפקודות PAPER, BORDER ו־BEEP (הנמצאות בשימוש במחשב זה Spectrum).

ספר BASIC זה פותח בחזרה על מספר נושאים שהוזכרו בספר המבוא, אך ביתר פירוט, תוך הדגשת הפעולות האפשריות לביצוע והבלתי נתנות לביצוע במחשב זה. כמו כן תמצא מספר תרגילים בסוף כל פרק. אל תתעלם מהם; רבים מהם מדגימים נקודות עליהן נרמז בכתוב־בפרק. בחן אותם ובצע את אלה המענינים אותך או את אלה המכסים את החומר שאינו נהיר לך דיו.

כל מה שלא תבצע, המשך להשתמש במחשב. באם אתה תוהה "מה תהיה תגובת המחשב באם אורה לו כך או כך?", אזי — התשובה לכך פשוטה בתכלית: הדפס זאת וצפה בתוצאה. בכל עת שספר זה מורה לך להדפיס משהו, שאל תמיד את עצמך: "מה ניתן להדפיס במקום זאת?" ונסה את תשובותיך. ככל שתרבה בכתיבת תכניות, כך תשפר את הבנתך לגבי פעולת המחשב.

בסוף ספר זה תמצא מספר נספחים. בין אלו תמצא הסברים על אופן ניהול הזכרון, ניהול המספרים במחשב וסדרת תכניות דוגמא להדגמת פעולת ZX Spectrum.

לוח המקשים:

אוסף התוים ב־ZX Spectrum כולל לא רק סימולים בודדים (אותיות, ספרות ועוד) אלא אף סימנים מורכבים (מילות מפתח, שמות פונקציות ועוד) כשכל אלו נקלטים מלוח המקשים ישירות — ללא צורך באיות שם הסימן־המורכב. (ע״י מספר מקשים) לשם כך — להשגת כל הפונקציות והפקודות הללו — למספר כפתורים קיימות חמש או יותר משמעויות שונות, המתקבלות בחלקן ע״י — שמוש בכפתורי ה־SHIFT (כלומר: לחיצה על CAPS SHIFT או על — SYMBOL SHIFT באותה עת שלוחצים על הכפתור המבוקש) ובחלקן — ע״י שימוש במחשב באופני פעולה שונים.

אופן הפעולה (mode) מסומן בעזרת סמן (cursor), אות מהבהבת המציינת להיכן תכנס התוית הבאה מלוח המקשים.

אופן הפעולה K (מהמילה Keywords — מילות מפתח) יחליף באופן אוטומטי את אופן הפעולה L כאשר המחשב יצפה לפקודה או לשורת תבנית (לעומת INPUT של מידע) וממקומו בשורה הוא יודע כי עליו לצפות למספר־שורה או למילת־מפתח. כך הדבר בתחילת כל שורה, או מיד לאחר THEN, או מיד לאחר: (מלבד במחרוזת — string). באם לא נעשה שימוש ב־SHIFT, המקש הבא יתפרש כמילת־מפתח (הכתובה על המקשים) או כספרה.

אופן הפעולה — L (מהמילה Letters — אותיות) מתרחש באופן נורמלי — בכל שאר הזמן. באם לא התבצע SHIFT, המקש הבא — יתפרש כסימן העקרי על אותו מקש, באותיות כתיב עבור אותיות. בשני האופנים, K וכן L, SYMBOL SHIFT ולאחריו מקש — יתפרשו כתוית הצדדית האדומה על המקש ו־CAPS SHIFT בצרוף עם — מקש ספרה יפורשו כפונקצית־הבקרה הכתובה בלבן מעל־המקש. CAPS SHIFT בצירוף עם מקשים אחרים אינה משפיעה על מילות־המפתח. באופן־פעולה K ובאופן הפעולה L היא הופכת האותיות לאותיות־דפוס.

אופן־הפעולה C (מהמילה Capitals — כותרות) הינו למעשה נוסח אחר לאופן־הפעולה L שבו כל האותיות מופיעות באותיות־דפוס. CAPS LOCK גורם למעבר בין שני אופני־פעולה אלו, מ־L ל־C ובחזרה.

אופן הפעולה E (מהמילה Extended — מורחב) משמש להשגת תוים נוספים, ברובם סימולים. הדבר מתבצע בעקבות לחיצה על שני כפתורי ה־SHIFT בו־זמנית — עבור לחיצה בודדת על מקש. באופן־פעולה זה האות פועלת כתוית או סימול (הנראה בירוק מעליו) באם לא בוצע SHIFT, ואחר (הנראה באדום מתחתיו) באם נלחץ בו זמנית על אחד מכפתורי ה־SHIFT. מקש ספרה פועל כסימול באם נלחץ ביחד עם SYMBOL SHIFT; אחרת הוא פועל כסדרת בקרת־צבע.

אופן פעולה G (מהמילה Graphic — גרפיקה) מתקיים בהקשת GRAPHICS (CAPS SHIFT עם 9), ונמשך עד אשר — הוקש שוב או 9 הוקש לבדו. מקש ספרה יביא לגרפיקת־פסיפס, יעצור את GRAPHICS או DELETE, וכל אחת ממקשי־האותיות מלבד Y, X, W, V וכן Z יביאו לגרפיקה שהוגדרה ע"י המפעיל.

באם מקש כלשהוא ילחץ באופן רצוף יותר מ־2 או 3 שניות, הוא יתחיל לחזור על עצמו ברציפות.
קלט מהמקשים מופיע במחצית התחתונה של המסך בעת שהוא מודפס, כשכל תוית (סימול יחיד או מורכב) מוכנסת לפני הסמן. נתן להזיז את הסמן שמאלה בעזרת CAPS SHIFT עם 5 או ימינה בעזרת CAPS SHIFT עם 8. התוית לפני הסמן נתנת להשמטה בעזרת DELETE (CAPS SHIFT עם Ø).
(הערה: נתן להשמיט את כל השורה בעזרת הדפסת EDIT (CAPS SHIFT עם 1) כשבעקבותיו — ENTER).

כאשר נלחץ המקש ENTER, השורה מבוצעת, מוכנסת לתכנית, או משמשת כ־INPUT של מידע בהתאם, אלא אם כן הוא מכיל שגיאת־תחביר (Sytax Error). במקרה זה ? מהבהב יופיע בסמוך לשגיאה.

עם הכנסת שורות התכנית, מוצג רישום במחצית העליונה של המסך. השורה האחרונה שהוכנסה — מכונה השורה־הנוכחית והיא מצוינת באמצעות הסימן >; נתן להזיזו בעזרת המקשים ↓ (CAPS SHIFT עם 6) וכן ↑ (CAPS SHIFT עם 7). באם נלחץ המקש EDIT (CAPS SHIFT עם 1), השורה הנוכחית מועברת לתחתית המסך שם היא נתנת לעריכה.

כאשר מבוצעת פקודה או כאשר תכנית "רצה", הפלט מוצג במחצית המסך־העליונה ונשאר שם עד אשר נכתבת שורת תכנית, או המקש ENTER נלחץ ביחד עם שורה ריקה, ↑ או ↓ בחלק התחתון מופיע דווח לפי קוד (אותיות או ספרות) המופיע בנספח II. הדווח נשמר במסך עד אשר נלחץ כפתור (המציין את אופן־הפעולה K).

בנסיבות מסוימות, CAPS SHIFT עם מקש SPACE מתנהג כמו BREAK, בעצרו את המחשב עם דווח D או L. זאת עבור:
(i) בסוף הצהרה בעת שהתכנית רצה, או
(ii) בעת שהמחשב משתמש בטייפ או במדפסת.

מסך הטלויזיה:

המסך כולל 24 שורות — באורך 32 תוים בכל אחת והוא מחולק לשני חלקים. החלק העליון כולל לכל היותר 22 שורות כשהתצוגה בו היא של רישום התכנית או פלט התכנית. כאשר ההדפסה בחלק־העליון הגיעה לתחתית קטע זה, "מגולגל" המסך שורה אחת כלפי־מעלה; באם הדבר כרוך באיבוד שורה אותה לא הפסקת לבחון, אזי המחשב עוצר ושואל — scroll? ("לגלגל") בלחיצה על N, SPACE או STOP תביא לעצירת התכנית עם דווח: D-BREAK-CONT; כל כפתור אחר יביא להמשך הגלגול.
החלק התחתון משמש לקליטת פקודות, שורות תכנית ומידע דרך — INPUT וכן — להצגת דווחים.
החלק התחתון מתחיל בקטע בן שתי שורות (העליונה — ריקה), אך הוא מתרחב כדי להתאים לקלט המודפס. כאשר הוא מגיע למיקום ה"שורה־הנוכחית" במחצית העליונה, הרחבות נוספות יביאו ל"גלגול" החלק העליון.

פרק 2: מושגי תיכנות — בסיסיים

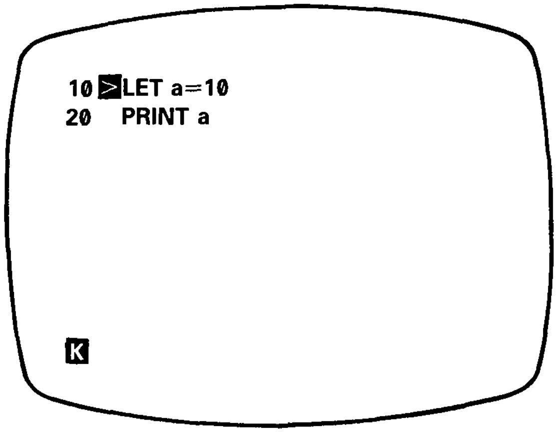
תמצית:

תכניות
מספרי שורות
עריכת תכניות בעזרת ↑, ↓ ו־EDIT.
LIST, RUN
PRINT, REM, NEW, INPUT, CONTINUE, GO TO
INPUT STOP
BREAK

הדפס את שתי השורות הבאות המהוות תכנית להדפסת סכומם של שני מספרים:

```
2Ø PRINT a
1Ø LET a=1Ø
```

כך שהמסך יראה כך:



כפי שכבר אתה יודע, מאחר ושורות אלו תחילתן עם מספרים, הן לא בוצעו מיידית — אלא נאגרו כשורות תכנית. לבטח שמת לב שמספרי השורות שולטים על סדר השורות בתכנית: הדבר חשוב במיוחד בעת הרצת התכנית, אך הוא בא לידי ביטוי בסדר השורות ברישום התכנית אותו אתה יכול לראות בעת על המסך.

עד כה הכנסת מספר אחד, לכן הדפס:

15 LET b = 15

וכבר היא בפנים. היה זה בלתי אפשרי להכניס שורה זו בין שתי השורות הקודמות באם הן היו ממוספרות 1 ו־2 במקום 1Ø ו־2Ø (מספרי השורות חייבים להיות מספרים שלמים בין 1 לבין 9999), וזוהי לכן הסיבה, מדוע בעת הדפסת תכנית לראשונה, כדאי מאד להשאיר מרווחים בין מספרי השורות.
כעת הנך צָרִיך לשנות את שורה 2Ø לשורה הבאה:

2Ø PRINT a+b

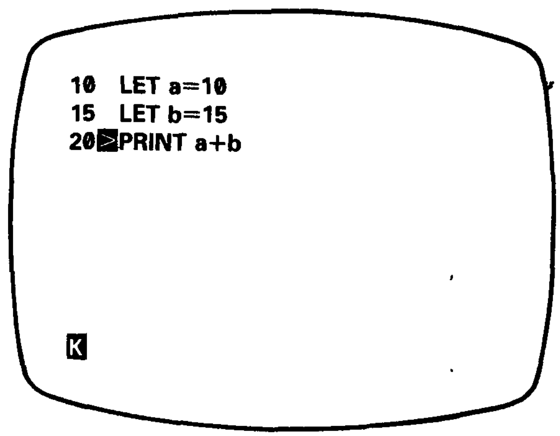
הנך יכול להדפיס את כל השנוי מתחילתו, אך פשוט יותר להעזר ב־EDIT שתוארה בספרון־המבוא. הסימול ⊠ בשורה 15 נקרא — סמן (cursor) — התכנית, והשורה אליה הוא מצביע כעת היא השורה — הנוכחית. זוהי בדרך־כלל השורה האחרונה אותה הדפסת, אולם אתה יכול להעזר במקשים ⇩ או ⇧ כדי להניע את הסמן מטה או מעלה, (נסה זאת, בהשאירך את סמן התכנית בסופו של דבר — בשורה 2Ø.)
בהקישך על המקש EDIT, העתק של השורה הנוכחית יוצג בתחתית המסך — במקרה זה, העתק של שורה 2Ø. החזק במקש ⇨ עד אשר הסמן L ינוע לקצה השורה, ואחר הדפס.

(ללא ENTER) + b

השורה בתחתית המסך תהְיה כעת

2Ø PRINT a + b

הדפס ENTER וזו תחליף את שורה 2Ø — הקודמת, כך שהמסך יראה כך:



הרץ תכנית זאת בעזרת RUN ו־ENTER והסכום יוצג. הרץ תכנית זו שוב ואחר הדפס:

PRINT a ,b

המשתנים נמצאים עדיין, למרות שהתכנית הסתיימה.
ישנה שיטה יעילה בהעזר ב־EDIT כדי להפטר מהחלק התחתון של המסך. הדפס אוסף של שטויות

(ללא ENTER) והחלט לאחר־מכן כי אינך רוצה בכך למעשה. דרך אחת להשמיט זאת היא לאחוז במקש DELETE עד אשר השורה תעלם; אך דרך אחרת הינה כדלהלן: באם תלחץ על EDIT, יוחלף כל הכתוב בתחתית בהעתק השורה הנוכחית. באם תלחץ כעת על ENTER, השורה הנוכחית תוחזר לתכנית ללא שנוי, תוך השארת תחתית המסך ריקה מתוכן.
באם הדפסת בטעות שורה, לדוגמא —

12 LET b = 8

היא תכנס לתכנית ואתה תווכח בטעותך. כדי להשמיט שורה בלתי־נחוצה זו, הדפס:

12 (כמובן עם ENTET)

לבטח תשתומם להווכח שסמן התכנית נעלם. דמיין לעצמך זאת — כאילו הוא נחבא בין שורות 10 ו־15, כך שאם תלחץ על ⇧ הוא ינוע למעלה לשורה 1Ø, ובאם תלחץ על ⇩ הוא ינוע למטה לשורה 15.
הדפס:

12 (וכן ENTER)

שוב, סמן־התכנית יחבא בין שורות 1Ø ו־15. כעת הקש EDIT ושורה 15 תופיע למטה: בעת שסמן התכנית נחבא בין שתי שורות, EDIT מביאה למטה את השורה הבאה שלאחר מספר השורה החדש.
הדפס ENTERכדי לנקות את החלק התחתון של המסך.
כעת הדפס:

3Ø (וכן ENTER)

כעת סמן התכנית יחבא לאחר קצה התכנית: באם תקיש EDIT, שורה 2Øתובא לתחתית המסך.
לבסוף, הדפס

LIST 15

תראה כעת על המסך:

15 LET b = 15
2Ø PRINT a + b

שורה 1Ø נעלמה מן המסך, אך היא עדיין בתוכניתך — ניתן להוכיח זאת ע״י הקשת ENTER.
ההשלכות היחידות של LIST 15 — הן יצירת רישום החל משורה 15, וקביעת — הסמן בשורה 15.
באם ברשותך תכנית ארוכה מאד, אזי LIST תהיה לבטח יעילה יותר להזזת סמן התכנית — מאשר ⇧ ו־⇩.
הדבר מדגים שימוש נוסף למספרי־שורה: הם מתנהגים כשמות לשורות התכנית כך שהנך יכול להתייחס אליהם, לעומת האופן בו למשתנים ישנם שמות.
LIST לבדו גורם לרישום החל מתחילת התכנית.
פקודה נוספת שראית בספרון־המבוא היא:

NEW

היא מוחקת תכניות קודמות ומשתנים קודמים במחשב. כעת הדפס בזהירות את התכנית הבאה, המשנה מעלות בפרנהייט לצלזיוס.

```
10 REM temperature conversion
20 PRINT "deg F", "deg C"
30 PRINT
40 INPUT "Enter deg F", F
50 PRINT F,(F−32)*5/9
60 GO TO 40
```

אתה תצטרך להדפיס את המילים שבשורה 1∅. כמו כן, למרות שבצירוף GO TO קיים רווח זוהי למעשה מילת־מקש אחת (על G).

כעת הרץ זאת. אתה תראה את הכותרות המודפסות על המסך על ידי שורה 2∅, אך מה קרה לשורה 1∅? כנראה המחשב התעלם ממנה לחלוטין. ובכן, אכן כך. REM שבשורה 1∅ פרושו REMark (הערה), והוא שם אך ורק כדי להזכיר לך מה התכנית מבצעת. פקודת REM מכילה REM שלאחריו כל דבר בו תחפוץ, והמחשב יתעלם מכך — עד לסוף השורה.

עד כה, המחשב הגיע לפקודת INPUT שבשורה 4∅ והוא ממתין שתדפיס ערך עבור המשתנה F — דבר אותו אתה יכול ללמוד מהעובדה שבמקום הסמן K לו ציפית קיים במקום סמן L. הכנס מספר; זכור את ENTER. כעת המחשב מציג את התוצאה והוא ממתין למספר נוסף. זאת עקב שורה 6∅ המכילה GO TO 4∅ — שמשמעותה בדיוק כאמור בה. במקום לבצע את התכנית ולהעצר, המחשב קופץ חזרה לשורה 4∅ ומתחיל מחדש. לכן הכנס טמפרטורה נוספת.

לאחר כמה שכאלו לבטח תתמה באם המכונה תשתעמם מכך אי־פעם; היא לא. בפעם הבאה שהיא תבקש מספר חדש, הדפס STOP. המחשב יענה עם דווח H STOP in INPUT in line 40: 1 שפרושו — הסיבה מדוע המחשב עצר והיכן (בפקודה הראשונה של שורה 4∅)

באם ברצונך להמשיך בתכנית הדפס —

CONTINUE

והמחשב יבקש ממך מספר חדש.

כאשר נעשה שימוש ב־CONTINUE המחשב זוכר את מספר השורה שבדווח האחרון אותו שלח לך, כל עוד שלא היה זה ∅ OK, וקופץ חזרה לאותה שורה: במקרה שלנו פירושו של דבר קפיצה לשורה 4∅, לפקודת — INPUT.

החלף את שורה 6∅ ב־GO TO 31 — לכך לא תהיה כל השפעה על מהלך התכנית. באם מספר השורה בפקודת GO TO אינו קיים, המחשב פונה לשורה הבאה שלאחר המספר הנתון. אותו הדבר מתקיים גם לגבי RUN; למעשה RUN לבדו פרושו RUN ∅.

כעת הדפס מספרים, עד אשר המסך יתמלא. כאשר הוא מלא, המחשב יעלה את כל המחצית העליונה של המסך שורה אחת כדי להכין מקום ריק, תוך איבוד תחילת המחצית העליונה. צורה זו נקראת "גלגול" — מסך (scrolling).

באם התעייפת מכך, עצור את התכנית תוך שימו ב־STOP והצג את רישום התכנית בעזרת ENTER.

בחן את הצהרת PRINT שבשורה 5∅. הפיסוק בה — הפסיק (,) — חשוב מאד, ואתה צריך לזכור שכאן הוא עוקב אחר חוקים מוגדרים בהרבה מאשר בפיסוק בשפת דיבור כגון אנגלית.

פסיקים גורמים להדפסה להתחיל או בשוליים השמאליים או בתחילת המסך, כתלות במי מופיע קודם. לכן בשורה 5∅, הפסיק גורם לטמפרטורה בצלזיוס להיות מודפסת במרכז השורה. עם נקודה־פסיק (;), מצד שני, המספר הבא או — המחרוזת הבאה — מודפס מיד לאחר הקודם. אתה יכול להווכח בכך בשורה 5∅, באם תחליף את הפסיק בנקודה־פסיק.

סימן פיסוק נוסף שבו אתה יכול להשתמש בפקודות PRINT הוא — הגֵרש ('). הוא גורם לכל שקיים בקרבתו להופיע בתחילת השורה הבאה במסך אך הדבר קורה בכל אופן בסוף כל פקודת PRINT, כך שלא תזדקק לגרש הרבה פעמים. זו הסיבה לכך שפקודת PRINT בשורה 5∅ תמיד מתחילה את ההדפסה בשורה חדשה, וגם הסיבה לכך שפקודת PRINT שבשורה 3∅ מייצרת שורה ריקה.

באם ברצונך למנוע זאת, כך שלאחר פקודת PRINT אחת — זו שלאחריה תמשיך באותה שורה, תוכל לשים פסיק או נקודה־פסיק בקצה הראשונה. כדי לראות כיצד הדבר פועל החלף שורה 5∅ בכל פעם באחת מהבאות:

```
5∅ PRINT F,
5∅ PRINT F;
```

וכן —

```
5∅ PRINT F
```

והרץ כל גרסה — ובאם תחפוץ נסה גם זאת:

```
5∅ PRINT F'
```

השורה עם הפסיק תביא להפרדת הכל לשתי עמודות, זו עם הנקודה־פסיק רוחסת את הכל יחד, זו ללא תוספת מביאה למספר בכל שורה ולכך גם מביאה זו עם הגרש — הגרש גורם לשורה חדשה מעצמו, אך מונע את האחת האוטומטית.

זכור את ההבדל בין פסיקים ונקודות־פסיק בפקודות PRINT; כמו־כן אל תבלבל אותם עם הנקודותיים (:) המשמשות להפרדת פקודות באותה־שורה.

כעת הדפס שורות נוספות אלו:

```
100 REM this polite program remembers your name
110 INPUT n$
120 PRINT "Hello ";n$;"!"
130 GO TO 110
```

זוהי תכנית נפרדת מקודמתה, אך אתה יכול לשמור את שתיהן במחשב באותה עת, כדי להריץ את החדשה הדפס —

RUN 1∅∅

מאחר והקלט בתכנית זו הינו מחרוזת (string) במקום מספר, היא מדפיסה שתי ציטטות — זאת כתזכורת־לך, ובכך חוסכת לך הדפסה. נסה זאת פעם עם כל כנוי בו תחפוץ לכנות את עצמך. גם בפעם הבאה, תקבל שוב שתי ציטטות, אך אינך חייב להשתמש בהן באם אינך חפץ. נסה זאת, כרוגמא. מחק אותן (עם 🡆 ועם DELETE פעמיים) והדפס:

n $

מאחר ואין כל מרכאות, המחשב יודע כי עליו לבצע חישובים: החישוב במקרה זה הוא למצוא את הערך של משתנה השרשרת שכנויו n $, שהוא השם אותו בחרת להכניס למחשב בסיבוב האחרון. כמובן, הצהרת INPUT פועלת כמו LET n $ = n $, כך שהערך של n $ נשאר ללא שנוי.

בפעם הבאה, לשם השוואה, הדפס שוב

n $

אך הפעם ללא מחיקת המרכאות. כעת, כדי לבלבלך, המשתנה n $ הוא בעל הערך "n $".

באם ברצונך להשתמש כמחרוזת קלט ב־STOP, עליך להזיז תחילה את הסמן לתחילת השורה בעזרת 🡄

כעת הבט שוב בפקודה RUN 1∅∅ שהיתה קודם. זוהי פשוט קפיצה לכתובת 1∅∅, אם כן האם לא יכולנו פשוט לרשום GO TO1∅∅? במקרה זה, התשובה הינה חיובית; אולם קיים הבדל. RUN 1∅∅ מוחקת תחילה את כל המשתנים ואת המסך ואחר כך פועלת בדיוק כמו GO TO 1∅∅. GO TO 1∅∅

אינה מוחקת דבר. ישנם מקרים שונים בהם תבקש להריץ תוכנית ללא איפוס משתנים; שם GO TO יהיה נחוץ — ו־RUN הרסני. לכן מוטב שלא נרגיל עצמנו להדפיס אוטומטית RUN כדי להריץ תכנית. הבדל נוסף הוא בכך שנתן להדפיס RUN ללא מספר שורה — והוא יתחיל מהשורה הראשונה בתכנית. GO TO חייב להופיע תמיד עם מספר שורה.
שתי תכניות אלו נעצרו מאחר והדפסת STOP בשורת־הקלט; לעתים — עקב טעות — אתה יכול לכתוב תכנית אשר אותה אינך יכול לעצור — ואשר לא תעצור בעצמה, אם כן הדפס:

```
2ØØ OG TO 2ØØ
RUN 2ØØ
```

זה נראה ערוך להמשיך עד אין קץ אלא אם תמשוך את תקע המתח מן השקע... אולם ישנה דרך פחות דרסטית: לחץ על CAPS SHIFT עם המקש SPACE — אשר מעליו כתוב BREAK. התכנית תעצור עם דווח L BREAK into program.
בסוף כל הצהרה, התכנית בודקת באם מקשים אלו נלחצים; בהתאם להם — היא עוצרת. המקש BREAK יכול גם לשמש כאשר אנו באמצע השימוש בטייפ־קסיטות או המדפסת, או מכונות שונות אחרות אותן אתה יכול לחבר למחשב — במקרה והמחשב מחכה שמכונות אלו יבצעו דבר כלשהוא אך הן לא מבצעות זאת.
במקרים אלו קיים דווח שונה D BREAK — CONT repeats. CONTINUE במקרה זה (ולמעשה ברוב המקרים גם כן) חוזר על ההצהרה בה נעצרה התכנית; אולם לאחר הדווח L BREAK into program, CONTINUE ממשיך הלאה עם ההצהרה הבאה תוך מתן אפשרות לקפיצות כלשהן להתבצע.
הרץ את תכנית השם, וכאשר היא מבקשת קלט — הדפס:

n $ (לאחר הסרת המרכאות)

n$ הינו משתנה בלתי מוגדר ולכן תקבל דווח שגיאה: 2 Variable not found. באם תדפיס כעת

```
LET n $ = "something definite"
```

(שלו דווח משלו 1 :Ø ,ØK Ø) וכן

```
CONTINUE
```

תווכח כי תוכל להשתמש ב־n$ במידע קלט ללא בעיה.
במקרה זה CONTINUE קופץ לפקודה INPUT שבשורה 11Ø. הוא מתעלם מהדווח מהצהרת LET מאחר וזה דווח 'OK', וקופץ חזרה לפקודה אליה התייחס הדווח הקודם, הפקודה הראשונה בשורה 11Ø. הדבר מיועד להיות שמושי. באם התכנית עוצרת בשל שגיאה כלשהיא, אתה יכול לנסות — דברים שונים כדי לתקן מצב זה ואז CONTINUE ימשיך לפעול לאחר־מכן.
כפי שציינו קודם־לכן, **הדווח** L BREAK into program הינו יחודי בכך שלאחריו CONTINUE אינו חוזר על הפקודה בה נעצרה התכנית.
הרישום האוטומטי (אלו שאינם תוצאה מפקודת LIST אך מתרחשים לאחר הכנסת שורה חדשה) יכול להביך אותך. באם תדפיס תכנית בת 5Ø שורות, שכולן הצהרות REM,

```
1 REM
2 REM
3 REM
:  :
:  :
49 REM
50 REM
```


אזי תוכל לנסות זאת.
הדבר הראשון שיש לזכור הוא שהשורה הנוכחית (עם ▷) תופיע תמיד על המסך בדרך כלל בקרבת — האמצע.
הדפס

LIST (וכמוכן ENTER)

וכאשר תשאל ?scroll ("לגלגל"?) — מאחר והמסך התמלא — לחץ n עבור לא (no). המחשב D BREAK — CONT repeats כאילו לחצת על BREAK. אתה עשוי במצב כלשהוא למצוא מה קורה באם תלחץ על y במקום n; n, SPACE וכן STOP נחשבים כ'לא' (No), בעוד שכל האחרים נחשבים כ'כן' (Yes).
כעת לחץ על ENTER פעם נוספת כדי לקבל רישום אוטומטי — כשאתה אמור לראות את שורות 1 עד 22 על המסך. כעת הדפס:

```
23 REM
```

ותקבל את שורות 2 עד 23; הדפס

```
28 REM
```

ותקבל את שורות 7 עד 28. (בשני המקרים, ע"י הדפסת שורה חדשה, הנעת את סמן התכנית כך שרישום חדש מתקבל.)
יתכן והדבר נראה שרירותי במידה מעטה. זהו למעשה נסיון לספק את רצונך, אף כי בני־האדם הנם יצורים שהתנהגותם אינה נתנת לחיזוי, אין זה תמיד מצליח — בנחוש כיאות.
המחשב שומר לעצמו רישום לא רק של השורה הנוכחית, זו שצריכה להופיע על פני המסך, אלא גם של השורה העליונה במסך. כאשר היא מנסה ליצור רישום־התכנית, הדבר הראשון אותו הוא מבצע — הינו להשוות את השורה העליונה לשורה הנוכחית.
באם השורה העליונה באה לאחר מכן, אזי אין כל טעם להתחיל בה, ולכן המחשב — משתמש בשורה הנוכחית כשורה עליונה חדשה — ומבצע את הרישום המתאים.
במקרים אחרים, השיטה היא להתחיל את הרישום מהשורה העליונה ולהמשיך ממנה עד לרישום השורה הנוכחית, תוך "גלגול" באם נדרש. בכל מקרה, תחילה מבוצע חישוב בכדי לבדוק מהו אורך הרישום שיתקבל, ובאם התשובה היא שיהיה זה ארוך מדי, אזי המחשב מניע את השורה העליונה כלפי מטה כדי לקרבה לשורה הנוכחית. כעת לאחר פעולתו על השורה העליונה, יתחיל המחשב את הרישום משם. באם, כאשר הוא יגיע לסוף התכנית או לתחתית המסך, השורה הנוכחית נרשמה, התהליך יעצר. בכל מקרה אחר, הוא "יגלגל" את המסך עד להופעת השורה הנוכחית על המסך, ולכל שורה נוספת אותה ירשום הוא יניע את השורה העליונה שורה אחת־מטה כך שהשורה העליונה תסחף בשכנות לשורה הנוכחית.
נסוי בהנעת השורה הנוכחית אפשרי בהדפסת

REM מספר־שורה

LIST מזיז את השורה הנוכחית אך לא את השורה העליונה, כך שרישומים מאוחרים יתכן ויהיו שונים. למשל הדפס

```
LIST
```

כדי לקבל את הרישום שיוצרת פקודת LIST ואחר הקש ENTER שוב, כדי להביא את שורה 1 לשורה העליונה. אתה אמור לקבל את שורות 1 עד 22 על המסך. הדפס

```
LIST 22
```

המציג את שורות 22 עד 43; כאשר תקיש שוב על ENTER, תקבל כתגובה את שורות 1 עד 22 על פני המסך. הדבר יעיל יותר עבור תכניות קצרות מאשר עבור ארוכות.
בהעזר בתכנית מלאת ה־REM שלמעלה, הדפס

LIST

ולאחר מכן n כאשר תשאל על־ידי המחשב ?Scroll. כעת — הדפס

CONTINUE

CONTINUE במקום זה הינו מהלך מחוכם, מאחר והחלק התחתון של המסך נהיה ריק; אולם אתה יכול לשוב ולשמור את המצב הרגיל בעזרת BREAK הסיבה היא ש־LIST היתה הפקודה הראשונה בשורה, ולכן CONTINUE חוזר על הפקודה. לרוע המזל, הפקודה הראשונה בשורה היא כעת CONTINUE עצמה ולכן המחשב פשוט נותר במצבו הנוכחי תוך ביצוע CONTINUE שוב ושוב עד אשר תעצור אותו.
תוכל לשנות זאת על ידי החלפת LIST בפקודה. LIST: שעבורה CONTINUE מגיב בדווח Ø OK (מאחרו־CONTINUE קופץ לפקודה השניה בשורה, אשר מהווה את סופה) או

::LIST

שעבורה CONTINUE מגיב בדווח N Statement lost (מאחר ו־CONTINUE קופץ לפקודה שלישית בשורה, שאינה קיימת).
עד כאן התוודעת להצהרות RUN, INPUT, LET, PRINT, NEW, CONTINUE, GO TO, LIST וכן REM, אשר כולן נתונות לשימוש הן כפקודות ישירות או בשורות תכנית — מצב המתקיים כמעט לגבי כל פקודות ה־BASIC במחשב — ZX Specturm. CONTINUE, LIST, RUN וכן NEW אינן בשימוש נרחב בתכניות אך גם הן נתונות לשימוש.

תרגילים:

1. הכנס הצהרת LIST בתכנית, כך שכאשר תריץ אותה היא תבצע LIST על עצמה.
2. כתוב תכנית שתקלוט מספרים ותדפיס את ניכויי המס. (שהינו 15%). הוסף הצהרות PRINT כך שהמחשב יכריז מה הוא עומד לבצע וישאל לערך המחיר — בנימוס רב. שנה את התכנית כך שתוכל להכניס את רמת המס (%). (כדי לאפשר שנויים בעתיד)
3. רשום תכנית שתדפיס סכום של מספרים אותם תכניס למחשב. (הצעה: העזר בשני משתנים שיקראו — total אשר יאופס בהתחלה — וכן item. הכנס את ערך item, סכמו עם total, הדפס את שניהם וחזור שוב על התהליך).
4. מה יוכלו CONTINUE וכן NEW לגרום בתבנית? האם תוכל למצוא לכך שמוש?

פרק 3: החלטות

תמצית:

STOP, IF
=, <, >, = <, = >, < >

כל התכניות אותן ראינו עד כה נתנו לחיזוי — ההוראות בהן בוצעו בצורה עוקבת ולאחר מכן חזרו שוב לתחילת התכנית. צורה שכזו איננה שמושית. באופן מעשי — קיימת הצפיה שהמחשב יחליט החלטות ויפעל בהתאם להן. ההוראה המשמשת זאת היא בצורה — IF (באם) משהוא נכון או לא THEN (אזי) בצע דבר־מה שהוא.
לדוגמא העזר בNEW כדי לנקות את הזכרון מהתכנית הקודמת, הדפס והרץ תכנית זו: (שמשמעותה בפשטות — משחק לשני שחקנים!)

```
10 REM Guess the number
20 INPUT a: CLS
30 INPUT "Guess the number", b
40 IF b=a THEN PRINT "That is correct": STOP
50 IF b<a THEN PRINT "That is too small, try again"
60 IF b>a THEN PRINT "That is too big, try again"
70 GO TO 30
```

אתה יכול להווכח שהצהרת (statment) IF הינה מהצורה —

IF תנאי THEN ....

כאשר "...." משמעותו סדרת פקודות, המופרדות ע"י פסיקים באופן הרגיל. התנאי הוא משהוא העתיד להתקיים כנכון או לא־נכון: באם התוצאה נכונה — אזי ההצהרה ביתרת השורה שלאחר ה־THEN מבוצעת, אולם בכל מקרה אחר תדלג התכנית על קטע זה והתכנית תבצע את השורה הבאה.
התנאים הפשוטים ביותר הנם בהשוואת 2 מספרים או מחרוזות: הם יכולים לבחון באם שני המספרים שווים או באם האחד גדול מן השני; הם יכולים לבחון באם שתי מחרוזות זהות או האם אחת מקדימה את השניה בסדר אותיות האלף־בית. הם משתמשים ביחסים =, <, >, < =, > = ו<>.
= פרוש שווה. למרות שזהו אותו סימון כמו בפקודת LET, כאן הוא משמש במשמעות שונה.
(SYMBOL SHIFT עם R) משמעותו: "קטן מ", כך שמתקיים:

1 < 2
—2 <—1
—3 < 1

כולם נכונים, אולם

1 < Ø
Ø < —2

אינם נכונים.

שורה 4Ø משווה את a ל־b. באם הם שווים אזי התכנית נעצרת ע״י פקודת STOP. הדווח בתחתית המסך 3 :3Ø, STOP, statment 9 מציין שההצהרה השלישית (או הפקודה) — בשורה 3Ø גרמה לתכנית להעצר — כלומר STOP.
שורה 5Ø קובעת באם b קטן מ־a ושורה 6Ø באם b גדול מ־a. באם אחד מתנאים אלו — נכון, ההערה המתאימה מודפסת והתכנית ממשיכה בפעולתה לשורה 7Ø המורה למחשב לשוב לשורה 3Ø ולהתחיל הכל מחדש.
הפקודה CLS (Clear Screen — נקה המסך) שבשורה 2Ø באה למנוע אפשרות מהשחקן השני לצפות בקלט שהכנסת למחשב.
אם כן.> (SYMBOL SHIFT עם T) פרושו: ״גדול מ־״ — בדומה ל־<, אולם הפוך במשמעותו. אתה יכול לזכור מי מביניהם הוא מה־בכך שתזכור שהקצה החד פונה תמיד למספר הקטן.
=> (SYMBOL SHIFT עם Q; אל תדפיס זאת כ־> שלאחריו = !) פרושו ״קטן מ־ או שווה לו״ והוא דומה אם כן ל־> מלבד העובדה שהוא נכון גם באם שני המספרים שווים: לכן 2=>2 נכון, אולם 2>2 אינו נכון.
=< (SYMBOL SHIFT עם E) פרושו ״גדול מ־ או שווה לו״ ומשמעותו בדומה לסימן <. <> (SHIFT SYBMOL עם W) פרושו — ״אינו שווה ל־״, המשמעות ההפוכה לסימן =.
המתמטיקאים רושמים בדרך־כלל את הסימנים=> <, = < וכן <> בצורה ≯, ≮, וכן # בהתאמה. הם רושמים גם בצורה הבאה: ״2<3<4״ שמשמעותו ״2<3 וכן 3<4״, אך הדבר אינו אפשרי בשפת BASIC.
הערה: בכמה גרסאות של BASIC — אך לא בזו של ZX-Spectrum הצהרת IF יכולה להיות מהצורה

IF תנאי THEN מספר־שורה

שמשמעותה בדיוק כמו —

IF תנאי THEN GO TO מספר־שורה

תרגילים:

1. נסה תכנית זו:

1Ø PRINT "X": STOP: PRINT "Y"

כאשר תריץ תכנית זו — היא תציג X ותעצור עם הדווח — 2: 1Ø, STOP statment 9. כעת הדפס

CONTINUE

אתה עשוי לצפות לקפיצה חזרה לפקודת ה־STOP — מאחר ופקודת CONTINUE חוזרת על ההצהרה אליה התייחס הדווח. בכל אופן, כאן הדבר לא יהיה יעיל, מאחר והמחשב פשוט יעצור שוב ללא הצגת Y. לכן — הדברים מסודרים כך שלאחר דיווח 9 — CONTINUE קופץ לפקודה שלאחר הפקודה STOP, כך שבדוגמא שלנו — לאחר CONTINUE, המחשב ידפיס Y ויגיע לסוף התכנית.

פרק 4: לולאות

תמצית:

NEXT, FOR
STEP, TO

נניח ואתה רוצה לקלוט במחשב 5 מספרים ולסכמם יחד. אפשרות אחת (אל תדפיס אותה — אלא אם כן אתה רואה זאת כחובה) היא לרשום

```
 10 LET total=0
 20 INPUT a
 30 LET total=total+a
 40 INPUT a
 50 LET total=total+a
 60 INPUT a
 70 LET total=total+a
 80 INPUT a
 90 LET total=total+a
100 INPUT a
110 LET total=total+a
120 PRINT total
```

שיטה זו אינה מהווה גישת תכנות טובה. היא נתנת לשליטה עבור חמישה מספרים, אולם אתה יכול לתאר לעצמך כמה מייגעת תראה תכנית שכזו לסיכום עשרה מספרים, ועד כמה בלתי־אפשרי יהיה לסכם מאה מספרים.
פשוט יותר להגדיר משתנה שימנה עד 5 ולאחר מכן יעצור את התכנית, באופן הבא (אשר אותו הדפס למחשב)

```
10 LET total=0
20 LET count=1
30 INPUT a
40 REM count=number of times that a has been input so far
50 LET total=total+a
60 LET count=count+1
70 IF count<=5 THEN GO TO 30
80 PRINT total
```

צורה זו של מנייה הינה כה שמושית עד כי ישנן שתי פקודות על מנת לפשטה: פקודת FOR ופקודת NEXT; הן נמצאות בשימוש תמיד יחדיו. בהעזר בהן התכנית אותה הדפסת זה עתה תבצע אותו הדבר כמו:

```
10 LET total=0
20 FOR c=1 TO 5
30 INPUT a
40 REM c=number of times that a has been input so far
50 LET total=total+a
60 NEXT c
80 PRINT total
```

(כדי לקבל תכנית זו מן הקודמת עליך לערוך, בלבד, את שורות 2∅, 4∅, 6∅, וכן 7∅. TO הינו SHIFT SYMBOL עם F).

שים לב ששינינו את Countלְ־C. משתנה המנייה — או משתנה הבקרה של לולאת FOR-NEXT חייב לכלול אות יחידה בשמו.

התוצאה בתכנית זו היא שְׁ־C משתנה בין הערכים 1 (ערכו־ההתחלתי) 2, 3, 4 וְ־5 (הגבול) — ולכל אחד שורות 3∅, 4∅ וְ־5∅ מבוצעות. כאשר C הגיע לערכו הגבולי — 5 שורה 8∅ מבוצעת.

נוסף לכך, משתנה הבקרה אינו נדרש להשתנות בקפיצות של 1 כל פעם: אתה רשאי לשנות 1 זה לכל ערך שתחפוץ על־ידי שמוש בְּ־STEP בפקודת הָ־FOR. הצורה הכללית ביותר של פקודת FOR הינה:

גודל השנוי STEP גבול TO ערך התחלתי = משתנה בקרה FOR

כאשר משתנה הבקרה הינו אות יחידה והערך ההתחלתי, הגבול וכן גודל הצעד הינם כל אותם דברים שהמחשב יכול לחשבם כמספרים — כגון: המספרים עצמם, סכומם או שמות משתנים מספריים. לכן, באם תחליף את שורה 2∅ שבתכנית בשורה

2∅ FOR C = 1 TO 5 STEP 3/2

אזי C ישתנה בין הערכִים 1, 2.5 וְ־4. שים לב שאין אתה צריך להגביל את עצמך למספרים שלמים וכן שמשתנה הבקרה אינו צריך לשוות ערך הגבול המדויק — הלולאה תמשך כל עוד שערכו קטן מערך הגדול או שווה לו.

נסה את התכנית הבאה המדפיסה את המספרים בין 1 לְ־10 בסדר הפוך

```
10 FOR n=10 TO 1 STEP -1
20 PRINT n
30 NEXT n
```

טענו קודם כי התכנית תמשיך בביצוע הלולאה כל עוד שמשתנה הבקרה קטן מערך הגבול או שווה לו. באם תבדוק את משמעות הדבר במקרה זה — תגלה כי התוצאה הינה שטות מוחלטת. יש, אם כן, לתקן את החוק הרגיל: כאשר הצעד הינו שלילי התכנית תמשיך בביצוע הלולאה כל עוד משתנה הבקרה גדול מערך הגבול או שווה לו.

עליך להיות זהיר בעת שימוש בשתי לולאות FOR-NEXT יחד, אחת בתוך השניה. נסה תכנית זו המדפיסה את המספרים עבור אוסף שלם של כלי דומינו — 6 נקודות:


```
10 FOR m=0 TO 6              }
20 FOR n=0 TO m        }     }
30 PRINT m;":";n;" ";  } n-loop  } m-loop
40 NEXT n              }     }
50 PRINT                     }
60 NEXT m                    }
```

אתה יכול להווכח שלולאת m הינה בשלמותה בתוך לולאה n מבנה הקרוי 'לולאות מקוננות' (nested loops). מצב שיש להמנע ממנו הוא המצב שבו שתי לולאות FOR-NEXT חופפות — כאשר אחת אינה במלואה בתוך השני, כגון:

```
5 REM this program is wrong
10 FOR m=0 TO 6        }
20 FOR n=0 TO m        }        }
30 PRINT m;":";n;" ";  } m-loop }
40 NEXT m              }        } n-loop
50 PRINT                        }
60 NEXT n                       }
```

שתי לולאות FOR-NEXT חייבות להיות או־אחת בתוך השניה או נפרדות לחלוטין.

דבר נוסף שיש להמנע ממנו הוא לקפוץ ממקום כלשהוא בתכנית מחוץ־ללולאה לתוך הלולאה. משתנה הבקרה משתנה כיאות רק כאשר מתבצעת הוראת — FOR ובאם תדלג עליה הוראת NEXT המתאימה תבלבל את המחשב. לבטח תקבל הודעת־שגיאה שתציין NEXT withovt FOR או Variable not found.

אין כל דבר שימנע בעדך מלהשתמש ב־FOR וְב־NEXT בפקודה ישירה. למשל נסה:

FOR m = ∅ TO 1∅ :PRINT m :NEXT m

אתה יכול לעתים להשתמש בכך כדרך (מלאכותית במידה מסוימת) לעקיפת המגבלה המונעת ממך להגיע לכל מקום שהוא באמצעות GO TO — עבור פקודה ללא מספר שורה. למשל:

FOR m=0 TO 1 STEP 0: INPUT a: PRINT a: NEXT m

גודל הקפיצה בערך ∅ כפי שכאן גורם לפקודה לחזור על עצמה — עד אין קץ.
גישה כזו אינה מומלצת, מאחר ואם צצה שגיאה אבדת את הפקודה ויהיה עליך להדפיסה שוב — וְ־CONTINUE לא יעזור.

תרגילים:

1. למשתנה בקרה אין רק שם וערך — כלמשתנה רגיל, אלא גם גבול, גודל צעד, ויחוס להצהרה שלאחר הצהרת הָ־FOR המתאימה. הוכח לעצמך כי כאשר הצהרת FOR מבוצעת — כל אותו מידע נתן להשגה (ע״י שימוש בערך ההתחלתי בערך הראשון אותו מקבל המשתנה) וכי מידע זה מספיק עבור הצהרת NEXT המתאימה כדי לדעת בכמה להגדיל את הערך, באם לקפוץ חזרה ואם כן לאיזו כתובת יש לקפוץ חזרה.

2. הרץ את התכנית השלישית שלמעלה ואחר הדפס —

PRINT C

מדוע התשובה הינה 6 ולא 5?
תשובה: הפקודה NEXT שבשורה 6Ø מבוצעת חמש פעמים ובכל פעם נוסף 1 ל־C. בפעם האחרונה C מקבל את הערך 6 ואז הפקודה NEXT מחליטה שלא להמשיך בביצוע הלולאה — אלא להמשיך הלאה בתכנית — כשערך C חלף מעבר לגבול שהוגדר לו.
מה יקרה באם תרשום STEP 2 בשורה 2Ø?

3. שַׁנה את התכנית השלישית כך שבמקום שתסכם אוטומטית חמישה מספרים, היא תשאל אותך למספר המספרים אותם — תרצה לסכם. כאשר תריץ תכנית זו, מה יקרה אם הקלט שתתן יהיה Ø — שפרושו שאינך רוצה לסכם מספרים כלשהם? מדוע אתה עשוי לצפות שהדבר יגרום לצרות במחשב למרות שכוונתך ברורה? (המחשב חייב לבצע חיפוש אחר הפקודה NEXT C, (שבדרך כלל אינו נחוץ) למעשה הדבר כולו נלקח בחשבון.

4. בשורה 1Ø של התכנית הרביעית שלמעלה, שנה את 1Ø ל־1ØØ והרץ את התכנית. היא תדפיס את המספרים בין 1ØØ ל־79 על פני המסך ולאחר מכן תשאל ?scroll בתחתית המסך. זאת כדי לתת בידך סכוי לראות את המספרים שמיד "יגולגלו" במסך מעבר לקצהו־העליון. באם תלחץ על n, STOP או BREAK — התכנית תיעצר בדווח: D BREAK — CONT repeats. באם תקיש במקש כלשהוא, יגרום מעשה זה להדפסת 22 שורות נוספות וישאלך שוב.

5. השמט את שורה 3Ø מן התכנית הרביעית. כאשר תריץ את התכנית החדשה, היא תדפיס את המספר הראשון ותעצור עם הודיעה Ø OK באם תדפיס

NEXT n

התכנית תמשיך פעם אחת בביצוע הלולאה ותדפיס את המספר הבא.

פרק 5: סברוטינות (תת־שגרות)

תמצית:

RETURN ,SUB

לעיתים, לחלקים שונים של התכנית — תפקידים דומים לביצוע, ואתה תמצא את עצמך מדפיס אותן שורות בשניים או יותר מקומות; אולם אין זה־הכרחי, אתה יכול להדפיס שורות אלו פעם אחת, בצורה הידועה כסברוטינה, ולאחר מכן להשתמש בה, או לקרוא לה — מבל מקום שהוא בתכנית ללא צורך להדפיסה שוב.
לביצוע זאת, עליך להשתמש בהצהרות המתאימות — GO SUB (שמשמעותה GO SUB routine — גש לסברוטינה) וְב־RETURN (חזור).
צורת הרישום היא:

GO SUB n

כאשר n הינו מספרה של השורה הראשונה בסברוטינה. צורה זו זהה בדיוק ל־GO TO n מלבד העובדה שהמחשב זוכר היכן הצהרת GO SUB היתה בתכנית כך שהוא יכול לשוב — לאחר ביצוע הסברוטינה. המחשב מבצע זאת ע"י הכנסת מספר השורה ומספר ההצהרה שבתוך השורה (יחדיו הן יוצרות את כתובת החזרה) בראש ערימה (הקרויה: מחסנית — Stack);

RETURN

נוטל את כתובת החזרה מן המחסנית ומחזיר את התכנית להצהרה שלאחריה.
כדוגמא הבה נבחון שוב את תכנית ניחוש המספרים. הדפס אותה — כמופיע בהמשך:

```
 10 REM "A rearranged guessing game"
 20 INPUT a: CLS
 30 INPUT "Guess the number ",b
 40 IF a=b THEN PRINT "Correct": STOP
 50 IF a<b THEN GO SUB 100
 60 IF a>b THEN GO SUB 100
 70 GO TO 30
100 PRINT "Try again"
110 RETURN
```

הפקודה GO TO בשורה 7Ø חשובה מאד, מאחר ואחרת התכנית תרוץ לתוך הסברוטינה ותגרום לשגיאה (7 RETURN without GO SUB) כאשר תגיע התכנית לפקודת ה־RETURN.
לפנינו תכנית נוספת, טפשית במקצת, המדגימה את השימוש בהצהרת GO SUB:

```
100 LET x=10
110 GO SUB 500
120 PRINT s
130 LET x=x+4
140 GO SUB 500
150 PRINT s
160 LET x=x+2
170 GO SUB 500
180 PRINT s
190 STOP
500 LET s=0
510 FOR y=1 TO x
520 LET s=s+y
530 NEXT y
540 RETURN
```

כאשר תכנית זו רצה, נסה לבחון באם אתה מבין מה מתרחש. הסברוטינה מתחילה בשורה 5ØØ. סברוטינה יכולה באושר רב לקרוא לסברוטינה אחרת, או אף לעצמה (סברוטינה הקוראת לעצמה היא סברוטינה רקורסיבית), כך שאל תחשוש מקיום מספר רב של רבדים.

פרק 6: READ, DATA, RESTORE:

תמצית:

READ, DATA, RESTORE

במספר תכניות קודמות ראינו כי אינפורמציה — או מידע (data), נתנים־להכנסה ישירות למחשב באמצעות ההצהרה INPUT. לעתים הדבר יכול להיות מאד מייגע, במיוחד באם מידע רב חוזר על עצמו בכל עת שהתכנית רצה אתה יכול לחסוך זמן רב בעזרת הפקודות DATA, READ ו־RESTORE. למשל

```
10 READ a,b,c
20 PRINT a,b,c
30 DATA 10,20,30
40 STOP
```

הצהרת READ מורכבת מ־READ שלאחריו רשימת שמות משתנים — המופרדים באמצעות פסיקים. הדבר פועל בדומה להצהרת INPUT מלבד העובדה שבמקום הדפסת הערכים המוקנים למשתנים, המחשב מקבל את הערכים מהצהרת DATA.
כל הצהרת DATA היא רשימת בטויים — מספריים או בטויי־מחרוזות המופרדים באמצעות פסיקים. אתה יכול למקמם בכל מקום בתכנית לפי רצונך, מאחר והמחשב מתעלם מהם מלבד כאשר הוא מבצע READ, דמה לעצמך את הבטויים מכל הצהרות ה־DATA בתכנית כמחוברים יחדיו ליצירת רשימה ארוכה אחת של בטויים, היא רשימת ה־DATA (DATA list). בפעם הראשונה בה המחשב נגש לביצוע READ של ערך כלשהו, הוא נוטל את הבטוי הראשון מרשימת ה־DATA; בפעם הבאה הוא יטול את השני; וכך — בכל התקלות בהצהרות READ — עוקבות, הוא יפעל באופן דומה לגבי כל רשימת ה־DATA. (באם הוא ינסה לקרוא מעבר לסוף הרשימה תתקבל שגיאה).
שים לב שזהו — בזבוז זמן לרשום הצהרות DATA בפקודות ישירות מאחר ו־READ לא ימצא אותן. הצהרות DATA יכולות להופיע רק בתכניות.
הבה נבחן כיצד מתאימות אילו יחדיו בתכנית שזה עתה הדפסת שורה 1Ø מורה למחשב לקרוא שלושה קטעי מידע ולהקנותם למשתנים a, b ו־c. שורה 2Ø מורה לבצע הדפסה של שלושת המשתנים. הצהרת ה־DATA שבשורה 3Ø נותנת את הערכים של a, b ו־c. שורה 4Ø עוצרת את התכנית. בכדי לראות את האופן בו מתבצעים הדברים — שנה את שורה 2Ø לצורה הבאה

```
20 PRINT b,c,a
```

האינפורמציה ברשימת ה־DATA יכולה להיות חלק מלולאת FOR-NEXT. הדפס:

```
10 FOR n=1 TO 6
20 READ D
30 DATA 2,4,6,8,10,12
40 PRINT D
50 NEXT n
60 STOP
```

כאשר תכנית זו רצה — אתה יכול לצפות בהצהרת ה־READ הנעה ברשימת ה־DATA. הצהרות DATA יכולות אף להכיל משתני־מחרוזות. לדוגמא:

```
10 READ d$
20 PRINT "The date is",d$
30 DATA "June 1st, 1982"
40 STOP
```

זוהי הדרך הפשוטה להבאת בטויים מרשימת ה־DATA: התחל בתחילת הרשימה ופעל דרכה עד לסיומה. בכל אופן, אתה יכול לגרום למחשב לנדוד ברשימת ה־DATA באמצעות פקודת ה־RESTORE. זו כוללת RESTORE שלאחריו מספר שורה, וגורמת להצהרות READ מאוחרות להתחיל לקבל את המידע מההצהרה הראשונה בשורה שמספרה נתון — או מזו שלאחריה. (אתה עשוי לוותר על ציון מספר השורה — ומקרה זה זהה למצב בו הדפסת את מספר השורה של השורה הראשונה בתכנית).
נסה תכנית זו:

```
10 READ a,b
20 PRINT a,b
30 RESTORE 10
40 READ x,y,z
50 PRINT x,y,z
60 DATA 1,2,3
70 STOP
```

בתכנית זו המידע הנדרש ע״י שורה 1Ø הביא למצב a=1 b=2. ההוראה: RESTORE 1Ø מאפסת את המשתנים ומאפשרת לבצע READ ל־x, y ו־Z כשההתחלה היא מהמספר הראשון בהצהרת ה־DATA.
הרץ תכנית זו שוב ללא שורה 3Ø ובחן את המתרחש.

פרק 7: בטויים

תמצית:

הפעולות +, —, *, /
בטויים, סימון מדעי, שמות משתנים

נוכחת עד כה בכמה מן הדרכים בהן ZX Spectrum יכול לחשב מספרים. הוא מסוגל לבצע את ארבע הפעולות האריתמטיות +, —, *, / (זכור כי * משמש להכפלה ו־/ לחילוק) והוא מסוגל למצוא את ערכו של משתנה בידיעת שמו.
הדוגמא —

LET tax = sum * 15 / 1ØØ

מרמזת על העובדה החשובה שחשובים אלו יכולים להיות מורכבים. מורכבות שכזו, כגון sum * 15/1ØØ נקראת בטוי (expression): אם כך בטוי בינו דרך תיאור קצרה להורות למחשב לבצע מספר חשובים, אחד אחר השני. בדוגמא שלפנינו, הבטוי sum * 15 / 100 פרושו — ׳בחן את ערכו של המשתנה הקרוי "sum", הכפל אותו במספר 15 וחלק התוצאה במספר 1ØØ׳
באם עד כה לא בצעת פעולות אלו, אנו ממליצים לך לעבור על ספרון־המבוא לשם לימוד האופן בו ZX Spectrum מטפל במספרים ובסדר בו הוא מחשב בטויים מתמטיים.

לסכום:

הכפלות וחלוקות מתבצעות תחילה. להן — עדיפות גבוהה על חיבור וחיסור, באופן יחסי לזה — לכפל או חילוק אותה עדיפות — כלומר הכפלות וחלוקות מתבצעות לפי סדר הופעתן משמאל לימין — בבטוי. לאחר מכן מתבצעים החיבורים וההחסרות, באם קיימים בבטוי, כשגם להם באופן יחסי זה לזה — אותה עדיפות, כך שסדר הביצוע הוא לפי סדר ההופעה — גם כן.
למרות שכל שנדרש ממך לדעת הוא — באם פעולה אחת היא בעדיפות מעל פעולה שניה, המחשב בודק זאת בעזרת מספר בתחום שבין 1 ל־16 המייצג את העדיפות של כל פעולה: הפעולות * ו־/ הן בעלות עדיפות 8 והפעולות + ו — בעלות עדיפות 6.
סדר זה של חישוב הינו קשיח לחלוטין, אולם אתה יכול לעקוף אותו על־ידי שמוש בסוגריים: כל מה שנמצא בתוך־סוגריים מחושב תחילה — ואחר נחשב כמספר יחיד.
בטויים הנם שמושים מאד — מאחר ובכל עת בה המחשב מצפה לקלוט ממך מספר, אתה יכול לספק לו בטוי במקום זאת — והוא יפעל למציאת התשובה, היוצאים מן הכלל הנם כה מעטים, עד כי נזכירם בדיון־במקרים המתאימים להם.
אתה רשאי לחבר יחד כמה מחרוזות שרק תחפוץ (או משתני־מחרוזות) בבטוי אחד, ובאם תחפוץ תוכל להעזר גם בסוגריים.
אנו חייבים לציין לך — כאילו שמות משתנים רשאי אתה להשתמש ובאילו אינך רשאי. כפי שכבר ציינו, שמו של משתנה — מחרוזת חייב להיות אות יחידה שלאחריה הסימן $; גם שמו של משתנה — הבקרה בלולאת FOR-NEXT חייב להיות אות יחידה; אולם שמות המשתנים המספריים מוגבלים פחות. אלו יכולים לכלול את כל האותיות והספרות כל עוד שהתו הראשון יהיה אות. אתה רשאי

לשים מרווחים בשם, כדי להקל על קריאתו, אך אלו לא יחשבו על־ידי המחשב כחלק מן השם. כמו כן, אין הבדל באם תדפיס את השם באותיות דפוס או אותיות כתיב.
לפנינו מספר דוגמאות לשמות משתנים המותרים במחשב:

```
X
t42
this name is so Long that I shall never be able to type it out again without making a mistake.
```

now we are six	(שני שמות אחרונים אלו נחשבים זהים,
nOWWeaRe SIX	ומתייחסים לאותו משתנה.)

השמות הבאים אסורים כשמות משתנים:

2ØØ1	(מתחיל עם ספרה)
3 bears	(מתחיל עם ספרה)
M * A * S * H	(* אינו ספרה או אות)
Fotherington — Thomas	(— **אינו ספרה או אות**)

בטויים מספריים נתנים ליצוג ע"י מספר ומעריך (אקספוננט): שוב, התייחס לספרון המבוא. נסה את הדוגמאות הבאות כדי שתובהר לך הנקודה:

```
PRINT 2.34 eØ
PRINT 2.34 e1
PRINT 2.34 e2
```

וכך הלאה עד —

```
PRINT 2.34 e15
```

תווכח כי לאחר זמן מה המחשב יתחיל בעצמו לרשום בסימון־מדעי. זה מאחר ולא יותר מארבע־עשרה תוים נתנים לשימוש כדי לכתוב מספר. נסה גם את —

```
PRINT 2.34e−1
PRINT 2.34e−2
```

וכך הלאה.
PRINT מאפשר הצגה — רק של 8 הספרות המשמעותיות ביותר של מספר. נסה

```
PRINT 4294967295, 4294967295−429e7
```

הדבר מוכיח כי המחשב יכול לשמור את ספרות המספר 4294967295 למרות שאינו מוכן להציג את כולן מיד.
בְ־ZX Spectzum נעשה שימוש באריתמטיקת — נקודה — צפה שפרושו — שהמחשב שומר בנפרד את ספרות המספר (המנטיסה) ואת מיקום הנקודה (האקספוננט־המעריך). הדבר אינו תמיד מדויק, אפילו עבור מספרים שלמים. הדפס:

```
PRINT 1e1Ø+1 — 1e1Ø, 1e1Ø — 1e1Ø+1
```

המספרים נשמרים בדיוק של כתשע וחצי ספרות, כך שהמספר 1e10 גדול מדי כדי להשמר בדיוק כפי שהוא־במקורו. אי הדיוק (למעשה בסדר גודל של 2) גדול מ־1, כך שהמספרים 1e10 וכן 1e10+1 נראים למחשב כשווים זה לזה.

להצגת דוגמא משונה יותר, הדפס:

```
PRINT 5e9 + 1 — 5e9
```

כאן אי־הדיוק של 5e9 הוא 1 בלבד והמספר 1 שיש לסכמו בהמשך מעוגל למעשה ל־2. המספרים: 5e9 + 1 וכן 5e9 + 2 נראים למחשב כשווים זה לזה.
השלם הגדול ביותר שנתן לשמרו מדויק־בשלמותו הוא המספר השווה ל־1 פחות מ־2 בחזקת 32 (או — 4,294,967,295).
המחרוזת "" — ללא כל תוים בתוכה נקראת "המחרוזת־הריקה" (null string). זכור כי למרווחים ישנה חשיבות, ומחרוזת־ריקה אינה זהה למחרוזת הכוללת בתוכה מרווחים בלבד.
נסה:

```
PRINI "Have you finished "Finnegans Wake" yet?"
```

כאשר תקיש ENTER יתקבל סימן־שאלה מהבהב המציין כי קיימת שגיאה במקום כלשהוא בשורה.
כאשר המחשב יגיע למרכאות בתחילת "Finnegans Wake" הוא חושב אותן כמציינות את סיום המחרוזת "Have you finished", ולכן אינו יכול לפענח את 'Finnegans Wake'.
כדי להתגבר על בעיה שכזו — בכל עת שתרצה לרשום מרכאות באמצע מהרוזת, עליך לרשמם פעמיים — כבדוגמא הבאה:

```
PRINT "Have you finished" "Finnegans Wake" "yet?"
```

כפי שאתה יכול ללמוד ממה שמודפס על־המסך, כל צמד מרכאות — מתקיימות למעשה פעם אחת; עליך רק להדפיסן פעמיים כדי שיוכרו על ידי המחשב כיאות.

פרק 8: מחרוזות

תמצית:

פריסה, בעזרת TO. שים לב שסימון זה אינו סטנדרטי ב־BASIC.

למחרוזת נתונה, תת־מחרוזת שלה כוללת כמה תוים עוקבים מתוכה שנלקחו בסדר עוקב. לכן "string" הינה תת־מחרוזת של "bigger string", אולם "b sting" וְ־"big reg" אינן כאלו.
קיימת צורת סימון הקרויה פריסה (slicing) לתיאור תת־מחרוזת, ונתן להשתמש בה לבטויי מחרוזת שונים הצורה הכללית היא:

(סיום TO התחלה) "בטוי־מחרוזת"

ולכן לדוגמא — מתקיים:

"abcdef" (2 TO 5) = "bcde"

באם תשמיט את ההתחלה, המחשב מניח שהכוונה למספר 1. באם תשמיט את הסיום המחשב מניח שהכוונה לאורך המחרוזת. לכן —

```
"abcdef"( TO 5)="abcdef"(1 TO 5)="abcde"
"abcdef"(2 TO )="abcdef"(2 TO 6)="bcdef"
"abcdef"( TO )="abcdef"(1 TO 6)="abcdef"
```

אתה יכול לרשום את האחרון גם בצורה: () "abcdef"
צורה שונה במקצת מתעלמת מ־TO וכוללת מספר אחד בלבד.

```
"abcdef"(3)="abcdef"(3 TO 3)="c"
```

למרות שבאופן־רגיל, גם ההתחלה וגם הסיום צריכים להתייחס לחלקים קיימים במחרוזת, כלל זה מבוטל על ידי כלל אחר: באם ההתחלה גדולה מהסיום, התוצאה היא המחרוזת־הריקה. לכן:

```
"abcdef"(5 TO 7)
```

יגרום לשגיאה subscyipt wrong, 3 כיון שהמחרוזת מכילה 6 תוים בלבד וְ־7 גדול מדי, אולם:

```
"abcdef"(8 TO 7)=""
"abcdef"(1 TO 0)=""
```

ערכי ההתחלה והסיום אסור שיהיו שליליים, אחרת תתקבל הודעת השגיאה B integer out of range.
התכנית הפשוטה הבאה מדגימה כמה מכללים אלו:

```
10 LET a$="abcdef"
20 FOR n=1 TO 6
30 PRINT a$(n TO 6)
40 NEXT n
50 STOP
```


הדפס NEW באם תכנית זו הודפסה על ידך, והכנס את התכנית הבאה:

```
10 LET a$="ABLE WAS I"
20 FOR n=1 TO 10
30 PRINT a$(n TO 10),a$((10-n) TO 10)
40 NEXT n
50 STOP
```

עבור משתני־מחרוזת, אנו יכולים לא רק לפרק תת־מחרוזות אלא גם להצמיד להן תת־מחרוזות. למשל — הדפס:

```
LET a$ = "I'm the ZX Spectrum"
LET a$ (5 TO 8) = "*******"
PRINT a$
```

שים לב כיצד, מאחר ותת המחרוזת a$ (5 TO 8) היא באורך 4 תוים, יעשה שימוש רק ב־4 הכוכבים הראשונים. זוהי תכונה אופינית של הצמדת תת־מחרוזות: תת־המחרוזות חייבות להיות בדיוק באותו הגודל אחר־כך כפי שהיו קודם. כדי להבטיח שכך באמת יתקיים, המחרוזת אותה מצמידים נקטמת בקצה הימני באם היא ארוכה מדי או מתמלאת ברווחים באם היא קצרה מדי — בדומה לשומר־הפונדק בסדום שדאג ל"התאים" את אורחיו למטתם על ידי מתיחת רגליהם או לחלופין קצוץ כפותיהם.
באם תנסה כעת

```
LET a$ () = "Hello there"
```

וכן —

```
PRINT a$
```

תווכח שאותו תהליך התרחש שוב (הפעם עם מרווחים) מאחר וְ־a$ נחשב כתת־מחרוזת.

```
LET a$ = "Hello there"
```

יבצע זאת כהלכה.
בטויי מחרוזת מסובכים ידרשו שימוש בסוגריים סביב להם בטרם יהיה אפשר לפרוס אותם. לדוגמא:

```
"abc" + "def" (1 TO 2) = "abcde"
("abc" +"def" ) (1 To 2) = "ab"
```

תרגיל:

1. נסה לרשום תכנית אשר תדפיס את היום בשבוע תוך שימוש בפריסת מחרוזות.
רמז: העזר במחרוזת Sun Mon Tues Wed Thur Fri Sat.

פרק 9: פונקציות

תמצית:

DEF
SQR ,INT ,ABS ,SGN ,VAL ,STR $,LEN
FN

הבה נבחן מכונה ליצור נקניק. אתה מכניס בקצה אחד שלה בשר, מסובב ידית ובקצה השני מתקבל נקניק. מבשר חזיר יתקבל נקניק חזיר, מבשר בקר — נקניק בקר וכו'.

פונקציות דומות להפליא למכונת־הנקניק, אלא שקיים כמובן הבדל: הן פועלות על מספרים ומחרוזות ולא על בשר. אתה מספק ערך אחד (הקרוי — ארגומנט), מבצע עליו מספר חישובים ומקבל ערך שני — התוצאה.

הכנסת בשר 🡄 מכונת נקניקים 🡄 קבלת נקניק
הכנסת ארגומנט 🡄 פונקציה 🡄 קבלת תוצאה:

ארגומנטים שונים מביאים לתוצאות שונות, ובאם הארגומנט אינו מתאים לגמרי — הפונקציה תעצור ותדווח על השגיאה.

כפי שקיימות מכונות שונות ליצור מוצרים שונים — אחת לנקניקים, אחרת ליצור סלטים — כך פונקציות שונות יבצעו חישובים שונים. לכל אחת ערך משלה — להבדילה משאר הפונקציות.

אתה תוכל להשתמש בפונקציות בבטויים על ידי הדפסת שמן כשלאחריו האגומנט, וכאשר הבטוי מחושב — מחושבת התוצאה של הפונקציה.

כדוגמא — נבחן את הפונקציה LEN, המספקת את אורכה של מחרוזת. הארגומנט שלה הוא המחרוזת שאת אורכה מבקשים לדעת והתוצאה היא האורך, כך שאם תדפיס

```
PRINT LEN "Sinclair"
```

תהיה תשובת המחשב — 8 שהוא מספר האותיות בשם 'Sinclair'. (כדי לקבל את הפונקציה LEN, כפי שגם קורה לגבי רוב שמות הפונקציות, עליך להעזר באופן־הפעולה־המורחב (Extended-Mode): לחץ על CAPS SHIFT עם SYMBOL SHIFT באותה העת כדי לשנות את הסמן מהצורה L לצורה E — ואז הקש על המקש K).

באם אתה מערב פונקציות ופעולות בבטוי אחד, אזי הפונקציות יחושבו לפני ביצוע הפעולות. יחד עם זאת, כמקודם, אתה יכול לעקוף כלל זה על ידי שימוש בסוגרים. לדוגמא — לפנינו שני בטויים הנבדלים רק בסוגרים, אך החשובים מבוצעים בסדר אחר לגמרי בכל מקרה (למרות שכך קורה — התוצאה הסופית זהה בשני המקרים).

LEN ("Fred" + bloggs")	LEN "Fred" + LEN "Bloggs"
LEN ("Fred bloggs")	4 + LEN "Bloggs"
LEN "Fred Bloggs"	4 + 6
1∅	1∅

לפנינו מספר פונקציות נוספות:

STR$ הפיכת מספרים למחרוזות: הארגומנט הינו מספר והתוצאה היא מחרוזת שתופיע על המסך באם המספר היה מוצג על ידי הצהרת PRINT. שים לב כיצד שמה מסתיים בסימן $ — כדי להזכיר שהתוצאה מפונקציה זו היא מחרוזת. לשם דוגמא, אתה רשאי לומר

```
LET a$ = STR$ 1 e 2
```

שלו אותה השלכה כמו הדפסת ההוראה

```
LET a$ = "1∅∅"
```

או — יכולת אף לומר,

```
PRINT LEN STR$ 1∅∅.∅∅∅∅
```

ולקבל את התוצאה 3 מאחר ו־"1∅∅" = STR$ 1∅∅.∅∅∅∅.

VAL דומה ל־STR$ אך הפוכה: הוא הופך מחרוזות למספרים. לדוגמא:

```
VAL "3.5" = 3.5
```

במובן מסוים, VAL היא ההפוכית של — STR$, מאחר ואם תקח מספר כלשהוא, תפעיל עליו את הפונקציה STR$ ולאחר מכן תפעיל את VAL — תקבל שוב את אותו מספר ממנו התחלת.

אולם, אם תקח מחרוזת, תפעיל עליה את VAL ואחר כך תפעיל על תוצאתה את STR$, לא תמיד תקבל את המחרוזת המקורית.

VAL הינה פונקציה "חזקה"־מאד, מאחר והמחרוזת המהווה את הארגומנט שלה אינה מוגבלת להיות דמוית מספר פשוט — היא יכולה להיות כל בטוי מספרי שהוא. לכן — לדוגמא

```
VAL "2 * 3" = 6
```

או גם —

```
VAL ("2" + "* 3") = 6
```

כאן מתבצעים שני תהליכים. בראשון, הארגומנט של VAL מחושב כמחרוזת: בטוי המחרוזת "3 *" + "2" מחושב כדי לתת את המחרוזת "3 * 2". לאחר מכן נמחקות מהמחרוזת המרכאות ומה שנותר מחושב כמספר: לכן 3 * 2 מחושב כדי לתת את התוצאה 6.

הדבר עשוי לבלבל מאד באם לא תפקח עינים; לדוגמא

```
PRI.NT VAL "VAL" "VAL""""2"""""""
```

(זכור כי בתוך מחרוזת — המרכאות צריכות להכתב פעמיים. באם תעמיק בתוך־המחרוזות תדרש להגדיל לארבעה מרכאות ואפילו שמונה).

קיימת פונקציה נוספת, דומה לפונקציה VAL, למרות שהיא פחות חזקה ממנה — הנקראת VAL $. המשתנה שלה הוא עדיין מחרוזת אך גם — התוצאה המתקבלת היא מחרוזת. כדי לראות כיצד היא פועלת זכור כיצד פועלת VAL בשני צעדים: תחילה מחושב הארגומנט כמחרוזת, לאחר מכן מוסרות מרכאות — המחרוזת וכל שנשאר מחושב כמספר. עבור VAL $ הצעד הראשון נשאר זהה, אולם לאחר שמרכאות המחרוזת הוסרו בצעד השני, כל שנותר מחושב כמחרוזת אחרת. לכן

```
VAL $ """Fruit punch""" = "Fruit punch"
```

(שים לב כיצד המרכאות משגשגות שוב)

```
LET a$ = "99"
```

והדפס את כל הבאים: VAL a$, VAL "a$", VAL"""a$""", VAL$ a$, VAL $ "a$" וכן VAL$"""a$""". כמה מאלו יתבצעו וכמה לא; נסה להסביר את כל התשובות. (שמור על קור־רוח)

SGN היא פונקצית־הסימן (sign). זוהי הפונקציה הראשונה שאינה קשורה למחרוזות, מבין אילו שראית, מאחר והן הארגומנט והן התוצאה הנם מספרים התוצאה היא 1+ באם הארגומנט חיובי, Ø באם הוא אפס ו־(1 —) באם הארגומנט שלילי.

ABS היא פונקציה נוספת אשר הארגומנט והתוצאה שלה הנם מספרים. היא הופכת את הארגומנט למספר חיובי (שהוא התוצאה) על ידי "שכיחת" הסימן, כך שלמשל —

ABS — 3.2 = ABS 3.2 = 3.2

INT פרושו integer — החלק השלם במספר, אפשרי גם מספר שלילי פונקציה זו הופכת מספר לא־שלם למספר שלם על ידי "זריקת" השבר שבמספר, כך שלדוגמא

INT 3.9 = 3

היה זהיר בעת השימוש בפונקציה זו עם מספרים שליליים, מאחר ופונקציה מעגלת את התוצאה תמיד כלפי מטה: לכן, לדוגמא

INT — 3.9 = -4

SQR מחשבת את השרש הרבועי (square-root) של מספר — כאשר התוצאה מוכפלת בעצמה נותנת את הארגומנט. לדוגמא

SQR 4 = 2 מאחר ו־2 * 2 = 4
SQR Ø.25 = Ø.5 מאחר ו־Ø.25 * Ø.25 = Ø.5
SQR2 = 1.4142136 (בקרוב) מאחר ו־1.4142136 * 1.4142136 = 2.ØØØØØØ1

באם תכפיל מספר כלשהוא (אפילו שלילי) בעצמו — התשובה תהיה תמיד חיובית. לכן — פרושו של דבר שלמספרים שליליים אין שורשים רבועיים, ולכן באם תפעיל את SQR על ארגומנט שלילי תקבל דווח שגיאה An Invalid Argument.

אתה יכול גם להגדיר פונקציות משלך. שמות אפשריים להן הם FN שלאחריו אות (באם התוצאה היא מספר) או FN שלאחריו אות המלווה בסימן $ (באם התוצאה הינה מחרוזת). בפונקציות אלו קיימת הקפדה על שימוש בסוגריים: הארגומנט חייב להופיע בתוך סוגרים.

תוכל להגדיר פונקציה בעזרת הצהרת DEF במקום כלשהוא בתכנית — לדוגמא, לפנינו הגדרת הפונקציה FNs שתוצאתה היא ריבוע הארגומנט:

1Ø DEF FN s(X) = X * X: REM the square of x

DEF מתקבלת גם באופן־הפעולה־המורחב (Extended Mode) בעזרת SYMBOL SHIFT עם 1. כאשר תדפיס זאת המחשב יציג אוטומטית FN, מאחר ובהצהרת DEF — FN עוקב תמיד לאחר DEF. לאחר זאת התו s משלים את שם הפונקציה — FNs.

ה־X שבסוגרים הוא השם שבעזרתו תוכל להתייחס לארגומנט, אתה רשאי להשתמש בכל אות יחידה שתרצה (או באם הארגומנט הינו מחרוזת — באות יחידה שלאחריה $).

לאחר הסימן = נמצאת הגדרת הפונקציה. זו יכולה להיות בטוי כלשהוא, והיא יכולה להתייחס לארגומנט על ידי שימוש בשם אותו כינית אותו (במקרה זה — X) כאילו היה זה משתנה רגיל.

כאשר הדפסת שורה זו, קראת לפונקציה זו כאילו קראת לאחת מן הפונקציות המוגדרות כבר במחשב, ע"י הדפסת שמה, FNs, כשלאחריו הארגומנט. זכור כי כאשר תגדיר פונקציה בעצמך, המשתנה חייב להופיע בתוך סוגרים. נסה זאת מספר פעמים:

PRINT FNs (2)
PRINT FNs (3 + 4)

PRINT 1 + INT FNs (LEN "chicken" / 2 + 3)

על ידי הכנסת הצהרת DEF מתאימה בתוך תוכניתך, תוכל להשתמש בפונקציות אותן הגדרת בבטויים שונים, באותה מידה של חופשיות כאילו השתמשת בפונקציות הקיימות כבר במחשב.

הערה: בכמה גרסאות של BASIC, אתה חייב אף לרשום את הארגומנטים של־מספר פונקציות הקיימות כבר במחשב בתוך סוגריים. אין הדבר כך בשפת BASIC של Zx-Specturm.

INT מעגלת תמיד כלפי מטה. כדי לעגל לשלם הקרוב ביותר — הוסף תחילה את הערך Ø.5. אתה יכול לבדך להגדיר פונקציה שתבצע זאת

2Ø DEF FN r(x) = INT (x+Ø.5): REM gives x rounded to the nearest integer.

תקבל — לשם דוגמא:

FNr (2.4) = 2 FN r(2.9) = 3
FNr (—2.4) = —2 FN r(—2.9) = —3

השווה תוצאות אלו עם התוצאות שיתקבלו לאחר שימוש ב־INT, במקום FNr. הדפס והרץ את התכנית הבאה:

```
10 LET x=0: LET y=0: LET a=10
20 DEF FN p(x,y)=a+x*y
30 DEF FN q()=a+x*y
40 PRINT FN p(2,3),FN q()
```

קיימות מספר נקודות בתכנית זו — שיש לתת עליהן את הדעת:

ראשית, פונקציה אינה מוגבלת לטפול בארגומנט אחד־בלבד: היא יכולה לטפל במספר ארגומנטים ומצד שני גם באף לא אחד, אולם עליך מוטל להמשיך ולרשום את הסוגרים.

שנית, אין זה משנה היכן תמקם בתכנית את הצהרות DEF. לאחר שהמחשב בצע את שורה 1Ø, הוא מדלג על שורות 2Ø ו־4Ø. הן חייבות להמצא במקום כלשהוא בתכנית. אין הן יכולות להמצא בתוך פקודה.

כמו כן, x ו־y הנם שמות המשתנים בתכנית כשלעצמה, ושמות הארגומנטים של הפונקציה FNp. FNp "שוכחת" באופן זמני מהמשתנים הקרויים x ו־y, אך מאחר ואין לה כל ארגומנט הקרוי a, היא "זוכרת" את המשתנה הקרוי a. לכן כאשר FNp (2, 3) מחושבת, a מקבל את הערך 1Ø מאחר והוא המשתנה, x את הערך הראשון מאחר והוא הארגומנט הראשון ו־y את הערך 3 מאחר והוא הארגומנט השני.

התוצאה הינה אז 1Ø + 2 * 3 = 16. כאשר FNq() מחושבת, אין כל משתנים, ולכן a, x ו־y עדיין יתייחסו למשתנים ויקבלו את הערכים הבאים 1Ø, Ø, Ø בהתאמה. התשובה במקרה זה תהיה 1Ø + Ø * Ø = 1Ø.

כעת שנה שורה 2Ø לנוסח הבא:

2Ø DEF FNp (x ,y) = FNq ()

הפעם ערך FNp (2, 3) יהיה 1Ø מאחר ו־FNq יחזור עדיין למשתנים x ו־y — במקום להשתמש בארגומנטים של — FNp.

בגרסאות BASIC מסוימות (לא בזו של ZX Spectrum) — קיימות פונקציות הקרויות LEFT $, RIGHT$, MID$ ו־TL$. LEFT$ (a$,n) — מייצרת את תת־המחרוזת של a$ המכילה את התוים הראשונים. (השמאליים).

RIGHT$ (a$,n) — מייצרת את תת־המחרוזת של a$ המכילה את אוסף התוים החל מהתו שמספרו n.
MID (a$, n1 ,n2) — מייצרת את תת המחרוזת של a$ המכילה n2 תוים המתחילים ברצף מהתו שמספרו n1.
TL$ (a$) — מייצרת את תת־המחרוזת של a$ המכילה את כל התוים מלבד הראשון.
תוכל להגדיר בעצמך פונקציות שיבצעו פעולות אלו, לדוגמא:

```
1Ø DEF FNt$ (a$) = a$ (2 TO) :REM TL$
2Ø DEF FNl$ (a$ ,n) = a$ (TO n) :REM LEFT$
```

בדוק שאלו פועלים עם מחרוזות באורך Ø ובאורך 1.
שים לב כי FNl$ כוללת 2 ארגומנטים — אחד מספר והאחר מחרוזת. פונקציה יכולה להכיל עד 26 ארגומנטים מספריים (מדוע 26?) ועד 26 ארגומנטים מחרוזתיים.

תרגיל:

השתמש בפונקציה FNs (X) = X*X כדי לבחון את הפונקציה SQR; אתה אמור להווכח כי מתקיים:

FNs (SQR X) = X

זאת באם תציג מספר כלשהוא במקום X, וכמו כן כי מתקיים

SQR FN s(X) = ABS (X)

עבור X חיובי או שלילי, (מדוע דרושה הפונקציה ABS?)

פרק 10: פונקציות מתמטיות

תמצית:

↑, PI, EXP, LN, SIN, COS, TAN, ASN, ACS, ATN.

פרק זה עוסק בפונקציות המתמטיות בהן יכול לטפל המחשב ZX-Spectrum. יתכן מאד שלא תזדקק להשתמש בפונקציה כלשהיא מבין אלו, ובאם תמצא זאת לנכון אל תחשוש לדלג על פרק זה. הוא מכסה את פעולת ↑ (העלאה בחזקה), הפונקציות EXP וכן LN והפונקציות הטריגונומטריות SIN, COS, TAN, והפונקציות ההופכיות להן ASN, ACS, ATN.

↑ וכן EXP:
אתה יכול להעלות מספר כלשהוא בחזקת מספר כלשהוא ופרושו של דבר — "הכפלת המספר הראשון בעצמו — כמספר פעמים השווה למספר השני", הדבר מוצג בקלות על־ידי רישום המספר השני — מעל לראשון ובצידו הימני; אך ברור שעקב מגבלות התצוגה — לא נתן לרשום על פני המסך ולכן ניעזר בסימן ↑ — במקום. לשם דוגמא, החזקות של 2 הנן:

2 ↑ 1 = 2	= 2	
2 ↑ 2 = 2 * 2	= 4	("2 בריבוע", שסימונו 2^2)
2 ↑ 3 = 2 * 2 * 2	= 8	("2 בשלישית", שסימנו 2^3)
2 ↑ 4 = 2 * 2 * 2 * 2	= 16	("2 ברביעית", שסימנו 2^4)

לכן, ברמה האלמנטרית ביותר, 'a↑b' פרושו 'a מוכפל בעצמו b פעמים', אך ברור שלדבר משמעות רק **אם** b הוא מספר שלם וחיובי. כדי למצוא את ההגדרה המתאימה לערכים אחרים של b, הבה נבחן את הכלל

a↑(b+c) = a↑b * a↑c

(שים לב כי לפעולה ↑ עדיפות על פני * או /, כך שכאשר קיימות מספר פעולות באותו בטוי, הפעולות ↑ מבוצעות לפני * ו־/). אינך זקוק להוכחה עבור ערכי b ו־c שהם חיוביים ושלמים; אך אם נחליט כי אנו מבקשים לנתח מצב שבו אינם כאלו — אני נמצא עצמנו חייבים לקבל את:

a↑0 = 1
a↑(—b) = 1/a   b
a↑(1/b) = השורש ה־b של a, שהוא — המספר שיש להכפילו בעצמו b פעמים לקבל a
a↑(b*c) = (a↑b)↑c

באם לא ראית עד כה בטויים כגון אלו, אל תנסה מיד ללמדם בעל־פה; זכור רק כי:

a↑(—1) = 1/a

וכן —

a↑(1/2) = SQR a

יתכן שכאשר תתוודע לאלו, יקבלו גם השאר משמעות — עבורך.
נסה אותם בעזרת התכנית הבאה:

```
10 INPUT a,b,c
20 PRINT a ↑ (b+c),a ↑ b*a ↑ c
30 GO TO 10
```

כמובן, שאם החוק אותו ציינו קודם — נכון, אזי בכל סיבוב שני המספרים אותם ידפיס המחשב יהיו שווים. (הערה — עקב צורת עבודת המחשב בביצוע פעולת ↑ , המספר שבצד שמאל — a במקרה זה, אסור שיהיה שלילי!)

דוגמא טפוסית לשימוש בפונקציה זו היא עבור חישוב רבית. הבה נניח כי אתה שומר את כספך בחברת־בניה המשלמת לך רבית 15% לשנה. אזי לאחר תום שנה אחת — יעמדו לרשותך לא רק 100% שהיו ברשותך בתחילת החסכון, אלא גם 15% שמשלמת לך חברת הבניה — כלומר115%, יחד, לעומת מה שהיה בידך בתחילת הדרך. במילים אחרות הכפלת את סכום השקעתך פי 1.15 (עובדה הנכונה ללא תלות בסכום עצמו אותו השקעת) לאחר שנה נוספת יעמוד לרשותך סכום הגדול פי 1.3225 = 2 1.15 = 1.15 * 1.15 מערך השקעתך. באופן כללי, לאחר y שנים יעמוד לרשותך סכום הגדול פי y 1.15 מהשקעתך הראשונית.

באם תנסה פקודה זו —

FOR Y = Ø TO 1ØØ :PRINT Y ,1Ø * 1.15 ↑Y :NEXT Y

אזי תווכח שגם הדפסה צנועה מתפתחת לסכום נכבד די במהירות וככל שהיא גדולה יותר, כך קצב גידולה מהיר יותר. (למרות שיתכן ותמצא שאין קצב זה עוקב אחר קצב מרוצת האינפלציה).

צורת התנהגות שכזו, שבה לאחר מרווח זמן קבוע, כמות מסוימת מכפילה את עצמה ביחס קבוע נקראת — "גידול אקספוננציאלי". הוא מחושב על ידי העלאת מספר קבוע בחזקת הזמן.

הבה נניח כי בצעת את:

1Ø DEF FN a(X) =a ↑X

כאן a הינו פחות או יותר קבוע בעזרת הצהרת LET: ערכו מתואם לשער הרבית המשתנה לעתים רחוקות. קיים ערך מסוים עבור a הגורם לפונקציה FN a לקבל צורה בעלת משמעות וצורה נאה בעיני מתמטיקאים: ערך זה קרוי e (בסיס הלוגריתמים הטבעיים). ZX Spectrum כולל פונקציה הקרויה:

EXP x = e ↑x

לרוע המזל, e עצמו אינו מספר כה "נאה": זהו מספר עשרוני אינסופי באורכו ולא מחזורי. תוכל לראות את המקומות הספורים הראשונים לאחר הנקודה בעזרת

PRINT EXP 1

מאחר וְ־EXP 1 = e ↑ 1 = e. כמובן, זהו רק קירוב. לעולם לא תוכל לרשום את e בדיוק מלא.

:LN

הפונקציה ההופכית לפונקצית האקספוננט היא הפונקציה הלוגריתמית: הלוגריתם (בבסיס a) של מספר x הוא החזקה שבה עליך להעלות את a כדי לקבל את המספר x; היא נרשמת בצורה $\log_a X$. לכן, לפי הגדרה $a \uparrow \log_a X = x$ וכן $\log (a \uparrow x) = x$.

לבטח אתה מכיר ויודע להשתמש בלוגריתמים עשרוניים לצורך ביצוע הכפלות; אילו קרויים לוגריתמים פשוטים. במחשב ZX Spectrum קיימת פונקציה LN המחשבת את הלוגריתם בבסיס a; אלו נקראים 'לוגריתמים טבעיים'. כדי לחשב לוגריתמים בכל בסיס אחר, עליך לחלק את הלוגרתמים הטבעיים בלוגריתם הטבעי של הבסיס:

$\log_a x$= **LN x/ LN a**

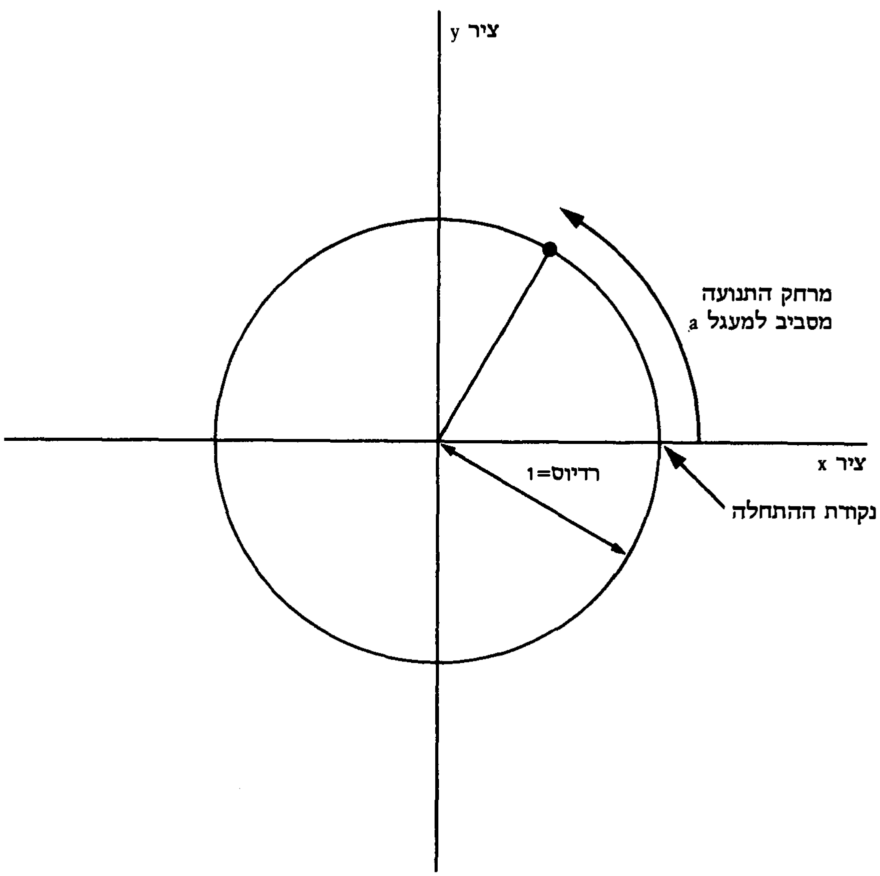
:PI

עבור מעגל נתון — היחס בין הקפו לבין קוטרו הוא מספר קבוע הקרוי Π. (Π הינו אות יוונית שקרויה pi).

בדומה למספר e, Π גם הוא מספר עשרוני, אינסופי באורכו ולא מחזורי; ערכו בדיוק מצומצם הינו — 3,141592653589.... המילה PI בְּ־ZX Spectrum (באופן פעולה המורחב ולאחר זאת M) מייצר מספר זה — נסה לבצע PRINT PI.

SIN, COS וכן TAN; ASN, ACS וכן ATN:

הפונקציות הטריגונומטריות בוחנות מה קורה כאשר נקודה נעה סביב למעגל. לפנינו מעגל ברדיוס — יחידה (איזו יחידה? אין זה חשוב כלל, כל עוד נשמור על אותן יחידות לכל אורך הדרך. אין כל דבר שימנע בעדך מלהמציא יחידות מדידה חרשות לכל מעגל־בו תהיה מעונין) ונקודה הנעה סביב לו. הנקודה החלה במקום 'השעה 3' ומשם נעה סביב בכוון נגד־סיבוב מחוגי־השעון.

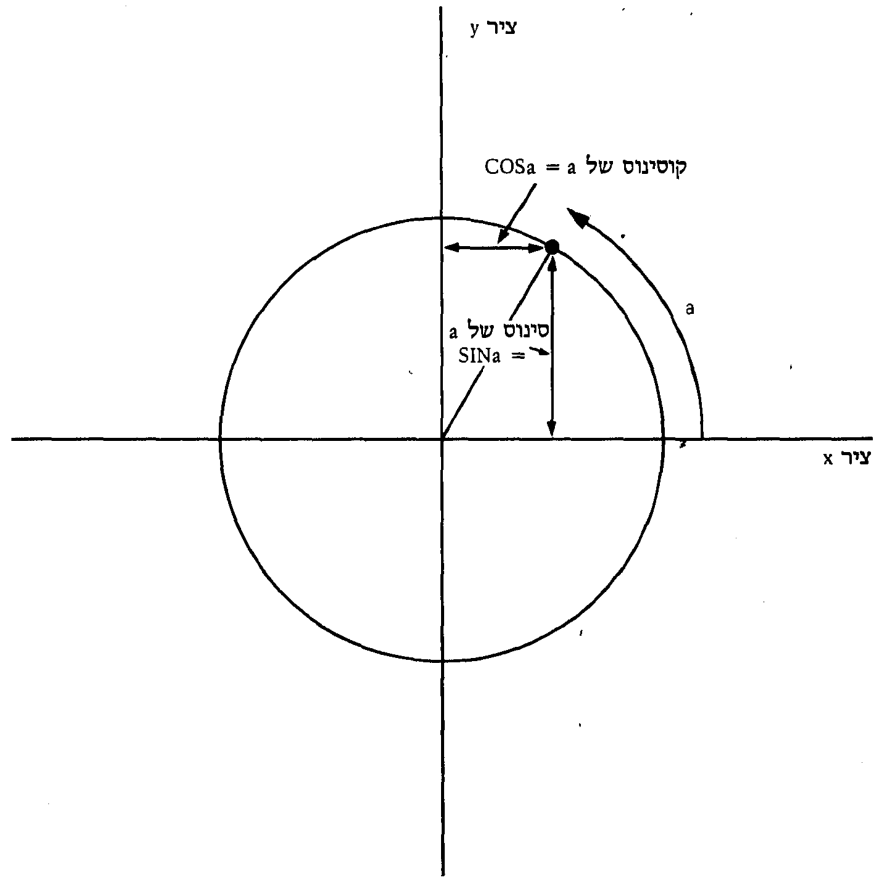


הוספנו בציור גם שני קוים הקרויים — 'צירים', דרך מרכז המעגל. האחד — בין שעה 3, ו'שעה 9' קרוי 'ציר X' והשני דרך 'שעה 6' וְ'שעה 12' קרוי 'ציר Y'.

כדי להגדיר את מקום הנקודה, עליך לציין כמה נעה מנקודת ההתחלה שב שעה 3: הבה נכנה מרחק זה a. אנו יודעים כי הקף המעגל הינו 2Π (מאחר ורדיוסו הינו כזכור 1 וקוטרו לכן 2): לכן כאשר היא נעה רבע מכלל הדרך שעַל הקף המעגל מתקיים: a = Π/2; כאשר הנקודה נעה מחצית ההקף אזי: a=Π; עבור סיבוב שלם: a=2Π.

עבור המרחק על ההקף a, הנתון בידינו, שני מרחקים נוספים המעוררים ענין — הם: המרחק מן הנקודה לציר Y והמרחק לציר X. אלו נקראים, בהתאמה, הקוסינוס (cos) והסינוס (sine) של a. הפונקציות COS וְ־SIN שבמחשב יחשבו אותן.

שים לב כי באם הנקודה עוברת משמאל לציר Y — אזי הקוסינוס (cos) הופך לשלילי; ובאם הנקודה עוברת מתחת לציר X אזי הסינוס (sin) הופך לשלילי.



תכונה נוספת היא שלאחר ש־a עבר את הערך 2Π, הנקודה חוזרת שוב לנקודה ממנה החלה והסינוס והקוסינוס שוב מתחילים מאותם ערכים:

SIN (a + 2 * PI) = SIN a
COS (a + 2 * PI) = COS a

הטנגנס של a (tg) מוגדר כמנת החילוק של sin בְּ־cos; הפונקציה המתאימה במחשב היא TAN. לעתים אנו זקוקים לחשב פונקציות אלו בכוון ההפוך, למצוא את ערך a שמקיים ערך סינוס מסוים, או ערך קוסינוס או טנגנס מסוים.

הפונקציות לביצוע זאת קרויות ארקסינוס (arc sin) — במחשב ASN, ארקקוסינוס (arccosine) — ACS במחשב וארקטנגנס (arctg) — במחשב ATN.

בדיאגרמה של תנועת הנקודה סביב למעגל, הבט ברדיוס המחבר את הנקודה למרכז המעגל. אתה אמור להבחין כי המרחק אותו קראנו a, המרחק אותו עברה הנקודה סביב להקף המעגל, הוא דרך למדידת הזוית בה נע הרדיוס מציר X. כאשר a=Π/2, הזוית היא 90 מעלות; עבור a=Π הזוית היא 180 מעלות; וכך־הלאה עד למצב a=2Π שבו הזוית 360 מעלות. אתה רשאי כמובן להתעלם מהמעלות, ולמדוד את הזוית במושגים של a בלבד; בכך, אנו טוענים, אנו מודדים את הזוית ביחידות 'רדיאנים'. לכן 2 Π רדיאנים שווים לְ־90 מעלות.

עליך לזכור תמיד כי בְּ־ZX Spectrum SIN, COS וכך הלאה משתמשות ברדיאנים ולא במעלות. כדי לתרגם מעלות לרדיאנים — חלק בְּ־180 והכפל בְּ־Π; כדי לתרגם חזרה מרדיאנים למעלות עליך לחלק בְּ־Π ולהכפיל בְּ־180.

פרק 11: מספרים אקראיים

תמצית:

RANDOMIZE
RND

פרק זה עוסק בפונקציה RND ובמילת־המקש RANDOMIZE. שניהם נמצאים בשימוש בהקשר למספרים־אקראיים, כך שעליך להיות זהיר בשימוש בהם כדי לא לבלבל ביניהם. שניהם נמצאים על אותו מקש (T); RANDOMIZE קוצר ל־RAND.

במובנים מסוימים RND דומה לפונקציה: הוא מבצע חישובים ומייצר תוצאה. הוא אינו רגיל, בכך־שאינו דורש ארגומנט.

בכל עת שתשתמש בו, תוצאתו תהיה מספר אקראי בתחום Ø עד 1. (לפעמים יתכן גם הערך Ø אך לעולם לא 1).

נסה את התכנית הבאה:

```
1Ø PRINT RND
2Ø GO TO 1Ø
```

כדי לבחון כיצד משתנה התשובה. האם אתה יכול לזהות תבנית כלשהיא בתוצאות שהתקבלו? אתה אמור שלא להיות מסוגל לעשות כן: 'אקראיות' פרושה — שאין כל תבנית — ברת זהוי.
למעשה, RND אינה פונקציה אקראית באמת, מאחר והיא משתנה בתחום מוגדר של 65536 מספרים. יחד עם זאת מספרים אלו "מעורבבים" היטב כך שאין שמץ של תבנית ברת־הבחנה, ולכן אנו אומרים כי RND היא פונקציה פסאודו־אקראית.

RND מספקת מספר אקראי בתחום Ø עד 1, אך אתה יכול בקלות להשיג מספרים אקראיים בתחומים אחרים. למשל RND*5 הוא מספר אקראי בתום Ø עד 5; 1.3+0.7*RND הוא מספר אקראי בתחום 1.3 עד 2. לקבלת מספרים שלמים העזר בפונקציה INT (זכור כי INT מעגלת תמיד כלפי מטה). כמו למשל 1+INT (RND*6) בה נשתמש בתכנית כדי לדמות קוביה. RND * 6 הוא בתחום Ø עד 6, אך מאחר ולמעשה מעולם לא מגיע לערך 6 כי (RND * 6) מקבל את הערכים Ø, 1, 2, 3, 4, ו־5.
לפנינו התכנית:

```
10 REM dice throwing program
20 CLS
30 FOR n=1 TO 2
40 PRINT 1+INT (RND*6);"  ";
50 NEXT n
60 INPUT a$: GO TO 20
```


הקש ENTER בכל עת שתרצה להטיל את הקוביה.
הצהרת RANDOMIZE משמשת לקבוע את המספר ההתחלתי — ממנו תתחיל פונקצית ה־RND בסדרת המספרים, כפי שתוכל להווכח מהתכנית הבאה:

```
10 RANDOMIZE 1
20 FOR n=1 TO 5: PRINT RND ,: NEXT n
30 PRINT : GO TO 10
```

לאחר כל ביצוע של RANDOMIZE 1, סדרת ה־RND תתחיל שוב עם 22735596 Ø Ø.Ø. אתה יכול להשתמש במספרים אחרים בין 1 ל־65535 בהצהרת RANDOMIZE כדי — להתחיל את סדרת RND במקומות השונים.
באם ברשותך תכנית הכוללת בתוכה RND והמכילה מספר שגיאות אותן לא מצאת, תבוא לעזרתך הצהרת RANDOMIZE כגון זו — כך שהתכנית תתנהג באותו אופן בכל עת שתריץ אותה.
RANDOMIZE לבדו (להצהרה RANDOMIZEØ — אותה משמעות) שונה, מאחר והוא מביא באמת ל־RND אקראי — כפי שתוכל לראות בתכנית הבאה:

```
1Ø RANDOMIZE
2Ø PRINT RND :GO TO 1Ø
```

הסדרה אותה קבלת — אינה אקראית לגמרי, מאחר ו־RANDOMIZE משתמשת בזמן שעבר מאז שהופעל המחשב. מאחר וערך זה גדל באותה מידה בכל עת שהצהרת RANDOMIZE מבוצעת, ה־RND הבא יתנהג בדומה לקודם. תקבל אקראיות טובה יותר ע"י החלפת GO TOØ בפקודה 2Ø GO TO.
הערה: ברוב הגרסאות של BASIC משתמשים ב־RND וב־RANDOMIZE ליצור מספרים אקראיים, אך לא כולם משתמשים בהם באותו האופן.
לפנינו תכנית ל'הטלת־מטבעות' ומניית מספר הפעמים שהופיע 'ראש' (head) או 'זנב ' (tail). ['עץ' או 'פלי' במונחים הלועזיים].

```
10 LET heads=0: LET tails=0
20 LET coin=INT (RND*2)
30 IF coin=0 THEN LET heads=heads+1
40 IF coin=1 THEN LET tails=tails+1
50 PRINT heads;",";tails,
60 IF tails<>0 THEN PRINT heads/tails;
70 PRINT : GO TO 20
```

היחס בין ה'ראשים' ל'זנבות' יהיה בקירוב 1 באם תמשיך במשחק זמן ממושך דיו, מאחר ובפעולה ממושכת אתה אמור לצפות למספר שווה בקירוב של 'ראשים' ו'זנבות'.

תרגילים:

1. בחן את הכלל הבא —
נניח ובחרת מספר בין 1 ל־872 והדפסת:

המספר־שבחרת RANDOMIZE

אזי הערך הבא של RND יהיה

(75 * (המספר שבחרת + 1) — 1) / 65536

2. (למתמטיקאים בלבד)
יהיה p מספר ראשוני (גדול) וְ־a שורש פשוט מודולו p. אזי אם bi היא השארית של ai מודולו p (1≤bi≤p-1) הסדרה:

bi—1

p—1

היא סדרה מחזורית של p—1 מספרים שונים בתחום Ø עד 1 (כולל 1) אשר — בבחירת a מתאים — נתן לגרום להם להראות אקראיים. 65537 הינו מספר (ראשוני) של פרמה — $2^{16} + 1$. מאחר והחבורה הכפלית של השאריות שאינן אפס מודולו 65537 היא מסדר שהוא חזקה של 2, השארית הינה שורש־פרימיטיבי אם ורק אם השארית אינה שארית מסדר רביעי. העזר בחוק־גאוס אודות ההדדיות מסדר ארבע כדי להראות כי 75 הוא שורש פשוט מודולו 65537.
הַ־ZX-Spectrum משתמש בגדלים p=65537 וכן a=75 ושומר כמה מן הַ־(bi-1) בזכרונו. RND מביא להחלפת bi-1 בזכרון בגודל bi+1—1). RANDOMIZE n (עבור 1≤n≤62535) משווה את bi לְ־n+1.
RND מפולגת בקירוב טוב — באופן אחיד על פני המרווח Ø עד 1.

פרק 12: מערכים

תמצית:

מערכים. (הדרך בה מטפל ZX Spectrum במערכי־מחרוזות אינה סטנדרטית במידת־מה).
.DIM...
הבה נניח שברשותך רשימת מספרים, לדוגמא ציוני עשרה אנשים בכתה. כדי לשמרם במחשב תגדיר משתנה יחיד עבור כל איש, אך תמצא זאת מכביד־מאד. אתה עשוי לקרוא למשתנים Bloggs 1, Bloggs 2 וכך הלאה עד Bloggs 10, אולם התכנית להגדרת משתנים אלו תהיה ארוכה ומשעממת להדפסה.
כמה יהיה זה באם היית יכול להדפיס

```
 5 REM this program will not work
10 FOR n=1 TO 10
20 READ Bloggs n
30 NEXT n
40 DATA 10,2,5,19,16,3,11,1,0,6
```

ובכן, אינך יכול,
עם זאת קיים מנגנון שבעזרתו נתן לישם רעיון זה — והוא בנוי על מערכים (arrays). מערך הינו אוסף של משתנים, המהווים את האלמנטים שלו, כולם בעלי אותו שם והנפרדים רק בעזרת מספר (התוית — label) הכתובה בסוגריים לאחר השם. בדוגמא שלנו השם יכול להיות b (כמו עבור משתני־הבקרה בלולאות FOR-NEXT, שם המערך חייב להיות אות בודדת) ועשרת המשתנים יהיו:
(1), b(2) ועד b(10).
האלמנטים של המערך מכונים: 'משתנים מותווים' — בניגוד ל'משתנים הפשוטים' שאותם כבר הספקת להכיר.
בטרם תוכל להשתמש במערך, עליך לשמור מקום מסוים עבורו בתוך המחשב, דבר המבוצע על ידי הצהרת DIM (dimeusion) – מימד.

DIM b (1Ø)

מגדירה מערך ששמו b והוא בעל מימד 1Ø (כלומר: הוא מכיל 1Ø 'משתנים מותווים' b(1), ..., b(1Ø)), ומאפסת את 1Ø האלמנטים שזה עתה הוגדרו. כמו כן משמיטה הצהרה זו כל מערך ששמו b שהתקיים קודם לכן. (אך לא־משתנה פשוט. מערך ומשתנה — מספרי פשוט באותו השם יכולים להתקיים יחדיו — ואסור שיווצר בלבול כלשהוא ביניהם מאחר ולמשתנה — מערך נוספת תמיד תוית).
התוית יכולה בעצמה להיות בטוי מספרי כלשהוא, כך שכעת אתה יכול להדפיס:

```
10 FOR n=1 TO 10
20 READ b(n)
30 NEXT n
40 DATA 10,2,5,19,16,3,11,1,0,6
```

אתה יכול להגדיר מערכים בעלי מימד גדול מאחד. במערך בעל שני מימדים תזדקק לשני מספרים כדי להגדיר אחד מן האלמנטים במערך — בדומה להגדרת מספרי השורה והעמודה — לשם הגדרת מיקום תוית ספציפית על פני מסך הטלויזיה — כך שלו תבנית של טבלה. לחלופין, באם תדמיין את מספרי השורה והעמודה (עבור שני מימדים) כמתייחסים לדף מודפס, תוכל להשיג מימד נוסף עבור מספרי הדפים. כמובן, אנו מדברים על מערכים מספריים; כך שהאלמנטים לא יהיו תוי־דפוס כמו בספר, אלא מספרים. חשוב על האלמנטים של מערך תלת מימדי־V כמוגדרים ע״י V (מספר הדף, מספר השורה, מספר העמודה)

לדוגמא, כדי להגדיר מערך דו־מימדי C בעל המימדים 3 ו־6, העזר בהצהרת DIM הבאה:

DIM C (3,6)

ובכך הגדרת מערך בן 18 = 6 * 3 'משתנים־מותווים':

c(1,1),c(1,2),...,c(1,6)
c(2,1),c(2,2),...,c(2,6)
c(3,1),c(3,2),...,c(3,6)

אותו עקרון מתקיים לכל מספר מימדיהם שהוא.

למרות שיתכן וברשותך יהיו מערך ומשתנה פשוט בעלי אותו שם, לא יתכן קיום שני מערכים בעלי אותו שם — גם אם מימדיהם שונים.

כמו כן קיימים מערכי־מחרוזות. המחרוזות במערך נבדלות ממחרוזות פשוטות — בכך שהן בעלות אורך קבוע — כ'מעשה סדום' — ע״י קיצוץ קצוותיהן או מתיחתן על־ידי הוספת מרווחים. דרך אחרת לראות אותם היא כמערכים (עם מימד נוסף) של תוים בודדים. שמו של מערך מחרוזת הוא אות יחידה שלאחריה הסימן $, כאשר מערך מחרוזת ומשתנה פשוט אסור שיהיו בעלי אותו שם (שלא כמו במקרה של מערך מספרי).

הבה נניח, שנדרש מערך a$ המכיל 5 מחרוזות. עליך להחליט כמה ארוכות יהיו מחרוזות אלו — הבה נניח שאורך של 10 תוים יענה על צרכינו. עליך להגדיר לאחר מכן

DIM as (5 ,10) (הדפס זאת)

בכך מוגדר מערך בן 50 = 10 * 5 תוים, אולם אתה יכול לראות כל שורה כמחרוזת:

a$(1)=a$(1,1) a$(1,2) ··· a$(1,10)
a$(2)=a$(2,1) a$(2,2) ··· a$(2,10)
: : : : :
a$(5)=a$(5,1) a$(5,2) ··· a$(5,10)

באם תציין את אותו מספר של תויות (2 במקרה זה) כמו בהצהרת DIM — תגדיר בכך תו יחיד; אולם באם לא תציין את התוית השניה תוגדר בכך מחרוזת שלמה בעלת אורך־קבוע. לכן, לדוגמא As (2,7) הינו האות ה־7 במחרוזת As (2). בהשתמש בשיטת הרישום כ'פרוסות', נוכל לרשום זאת גם בצורה — As (2) (7). הדפס כעת:

וכן —

```
LET As (2) = "123456789Ø"
PRINT as(2) ,a$ (2 ,7)
```

ותקבל —

```
123456789Ø   7
```

עבור התוית האחרונה (זו שאתה תשמיט), אתה יכול גם לדאוג ל'פריסה לפרוסות', כך שלמשל:

as (2 ,4 TO 8) = as (2) (4 TO 8) = "45678"

זכור:

במערך מחרוזות לכל המחרוזות אותו אורך קבוע. להצהרה DIM נוסף מספר (האחרון) להגדרת אורך זה. כאשר אתה כותב 'משתנה מותווה' עבור מערך־מחרוזות, אתה רשאי לרשום מספר נוסף, או 'פורס' כדי שיתאים למספר הנוסף בהצהרת DIM.

תוכל למצוא גם מערכי־מחרוזות ללא כל מימד. הדפס

DIM as (1Ø)

ואז תגלה כי a$ מתנהג בדיוק כמו משתנה מחרוזת, מלבד העובדה שהוא תמיד באורך 1Ø — בגישת 'מיטת סדום', כפי שצוין כבר קודם.

תרגילים:

1. בעזרת ההצהרות READ ו־DATA הגדר מערך m$ המכיל 12 מחרוזות כאשר ms (n) הוא שם החודש ה־n־. (רמז: ההצהרה DIM תהיה DIM ms (12 ,9). נסה זאת על־ידי הדפסת כל האלמנטים ms (n). (העזר בלולאה)
הדפס:

PRINT "now is the — month of" ;ms (5) ;"ing" ;"when merry lads are playing"

מה תוכל לעשות ביחס לכל אותם המרווחים?

פרק 13: תנאים

תמצית:

OR ,AND
NOT

ראינו בפרק 3 כיצד הצהרת IF נראית:

IF תנאי THEN...

התנאים שם היו היחסים (=, >, <, => ,=< ,<>) המשווים שני מספרים או שתי מחרוזות. אתה יכול לאחד כמה מהן בעזרת הפעולות הלוגיות AND, OR וְ־NOT.
בטוי כלשהוא AND בטוי אחר — הינו 'נכון' רק בעת ששני הבטויים הנם 'נכונים', כך שברשותך יכולה להיות שורה מהצורה:

```
IF a$ = "yes" AND x > Ø THEN PRINT X
```

שבה x יודפס באם "a$="yes וכן x>Ø. שפת ה־BASIC, בנושא זה, קרובה לשפה האנגלית המדוברת — כך שדומה, כמעט, שאין טעם לפרט. כמו באנגלית, אתה יכול לצרף־יחד אוסף של קשרים בעזרת AND כאשר הצרוף כולו יהיה 'נכון' — באם המרכיבים יהיו כולם 'נכונים'.
בטוי כלשהוא OR בטוי אחר הינו 'נכון' בכל מקרה — כשלפחות מתקיים שאחד משני הבטויים 'נכון'. (זכור כי הוא 'נכון' גם באם שניהם 'נכונים'; זהו מצב שאינו תואם תמיד עם האנגלית).
הבטוי NOT הופך בטויים מצורה אחת להפוכה לה. NOT — הבטוי הינו 'נכון' באם הבטוי עצמו 'אינו־נכון', ו'אינו־נכון' כל עוד שהבטוי הינו 'נכון'.
נתן לבטא בטויים לוגים בעזרת AND, OR, וְ־NOT כשם שבטויים מספריים נתנים לבטוי בעזרת מספרים והפעולות +, —, * וכו'; אתה רשאי אף להשתמש בסוגרים בעת הצורך. גם עבור הפעולות הלוגיות קיימות עדיפויות בדומה לפעולות +, —, / וכן ↑: OR היא בעלת העדיפות הנמוכה ביותר, לאחריה AND ולבסוף NOT — ולאחר מכן הבטויים והפעולות הרגילות.
NOT היא למעשה פונקציה, עם ארגומנט ותוצאה, אולם העדיפות שלה נמוכה מאד ביחס לפונקציות אחרות. לכן הארגומנט בה אינו דורש שימוש בסוגרים — אלא אם כן הוא מכיל AND או OR (או שניהם). NOT a = b פרושו זהה בדיוק לבטוי הלוגי NOT (a=b) (וכמובן שכך מתקיים גם עבור <>).
<> הוא השלילה של = במובן שהוא 'נכון' אם ורק אם = 'אינו־נכון'. במילים אחרות

a <> b זהה לבטוי NOT a=b

וכמו כן —

NOT a<>b זהה לבטוי a=b

הוכח לעצמך כי =< וכן => הם השלילות של < וכן > בהתאמה: לכן אתה יכול להפטר מ־NOT שבתחילתו של בטוי כלשהוא ע"י שנוי הקשר.
כמו כן —

NOT ([בטוי לוגי אחד] AND [בטוי לוגי שני])

זהה בדיוק לבטוי —

(NOT (הבטוי הראשון) OR NOT (הבטוי השני

בהעזר באלו אתה יכול לפתח את בטוי ה־NOT באמצעות סוגריים עד שלבסוף יושמו כולם על יחסים ותוכל אז להפטר מהם. לוגית, NOT הינו בלתי־נחוץ, למרות שיתכן ותמצא כי השימוש בו גור לתכנית להראות 'יפה' יותר.
הקטע הבא מסובך קמעא ונתן בפרוש לדלג עליו באם אתה בעל לב־חלש!
נסה —

```
PRINT 1 = 2 ,1 <> 2
```

שאמור להביא להדפסת שגיאת־תחביר. למעשה, עד כמה שהדבר נוגע למחשב לא קיים ערך לוגי: במקום זאת הוא משתמש במספרים רגילים, כפופים למספר כללים.
(i) =, <, >, =<, => וכן <> כולם מייצרים מספר כתוצאה: 1 עבור 'נכון' וְ־Ø עבור 'לא־נכון'. לכן, פקודת הָ־PRINT שלמעלה תדפיס Øעבור '1=2' — 'שאינו נכון' וְ־1 עבור '1<>2' — שהינו 'נכון' (ii) בהצהרה —

IF תנאי THEN ...

התנאי יכול למעשה להיות כל בטוי מספרי שהוא. באם ערכו Ø אזי הוא יחשב כְ'לא־נכון' ואילו כל ערך אחר (כולל הערך 1 שקשר 'נכון' יתן) יחשב כ'נכון'. לכן להצהרת IF אותה משמעות כלהצהרה הבאה —

IF תנאי <> Ø THEN ...

(iii) AND, OR וכן NOT גם הן פעולות המיצרות ערכים מספריים.

X AND Y מקבל את הערך X באם Y 'נכון' (שונה מִ־Ø)
Ø ('לא נכון') באם Y 'אינו־נכון' (Ø)

X OR Y מקבל את הערך 1 ('נכון') באם Y 'נכון' (שונה מִØ)
X באם Y 'אינו־נכון' (Ø)

NOT X מקבל את הערך Ø ('לא־נכון') באם X 'נכון' (שונה מִ־Ø)
1 ('נכון') באם XØ 'אינו־נכון' (Ø)

(שים לב כי 'נכון' פרושו שונה מִ־'Ø' כאשר אנו בודקים ערך נתון, אך פרושו '1' כאשר אנו מייצרים אחד חדש).
קרא שוב פרק זה — לאור נקודות אלו, והבטח לעצמך כי הכל ברור.
בבטויים x AND y, x OR y, וכן NOT x, x וכן y יקבלו בדרך־כלל את הערכים Ø וְ־1 עבור 'לא־נכון' ו'נכון' בהתאמה. עבור 10 הקומבינציות האפשריות (4 עבור AND, 4 עבור OR וְ־2 עבור NOT) ובדוק כי אכן הן מקיימות את שפרק זה מנסה לשכנע אותך שהן מבצעות.
נסה את התכנית:

```
10 INPUT a
20 INPUT b
30 PRINT (a AND a>=b)+(b AND a<b)
40 GO TO 10
```

בכל פעם היא מדפיסה המספר הגדול מבין שני המספרים a וְ־b.
הוכח לעצמך כי אתה יכול לחשוב על

x AND y

כבעל המשמעות

x באם y (אחרת התוצאה תהא Ø)

וכן — על הבטוי

x OR y

כבעל המשמעות

x אלא־אם כן y (בכל מקרה התוצאה תהיה 1)

בטוי המשתמש בְ־AND וְ־OR כגון אלו, נקרא "בטוי — מותנה" (conditional expression). דוגמא, בעזרת OR יכולה להיות: LET total price = price less tax * (1.15 OR V$ = "zero rated")
שים לב כיצד AND נוטה ללכת עם חיבור (מאחר וערך 'ברירת־המחדל' שלה הוא Ø) וְ־OR נוטה ללכת עם כפל (מאחר וערך 'ברירת־המחדל' שלה הוא 1).
אתה יכול לבצע בטויי־מחרוזות מותנים, אולם רק עם פעולת AND.

xs AND y מקבל הערך
- x$ באם y שונה מאפס
- "" באם y שווה לאפס.

כך שמשמעותו: x$ באם y (אחרת — ה'מחרוזת הריקה')
נסה את התכנית הבאה, המקבלת שתי מחרוזות ומסדרת אותן לפי סדר האלפא־בית:

```
10 INPUT "type in two strings"'a$,b$
20 IF a$>b$ THEN LET c$=a$: LET a$=b$: LET b$=c$
30 PRINT a$;" ";("<" AND a$<b$)+("=" AND a$=b$);" ";b$
40 GO TO 10
```

תרגיל:

1. BASIC יכולה לעבד שורות. מהשפה האנגלית לבדה. קח לדוגמא את המשפט האנגלי — 'באם a אינו שווה לְ־b או c'. כיצד תרשום זאת בשפת BASIC?
התשובה אינה מהצורה —

```
IF A <> B OR C
```

וכמו כן גם לא —

```
IF A<>B or a<>C
```


פרק 14: אוסף התוים

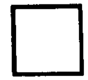
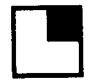
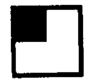
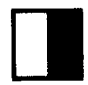
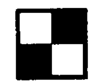
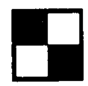
תמצית:

CODE ,CHR$
POKE ,PEEK
USR
BIN

האותיות, הספרות, סימני־הפיסוק וכל יתר הסימנים היכולים להופיע במחרוזות, נקראים תוים (characters), והם מרכיבים את ה'אלף־בית' — או 'אוסף התוים' בו משתמש ZX Spectrum. רוב התוים הללו הם סימולים (symbols) בודדים, אך ישנם כמה שהם צרופים (tokens) המייצגים מילים שלמות כגון PRINT, STOP, <> ועוד.
בפועל קיימים 256 תוים ולכל אחד קוד בין Ø לְ־255. רשימה מלאה שלהם תמצא בנספח A. כדי לתרגם בין קודים לתוים, נעזר בשתי הפונקציות CODE וְ־CHR$.
CODE פועלת על מחרוזות ומייצרת את הקוד המתאים לתו — הראשון במחרוזות (או Ø באם המחרוזות ריקה).
CHR$ פועלת על מספר ומייצרת מחרוזות בת תו־בודד אשר הקוד שלה מתאים למספר הזה.
התכנית הבאה מדפיסה את כל אוסף התוים:

```
1Ø FOR a = 32 TO 255 :PRINT CHR$ a;:NEXT a
```

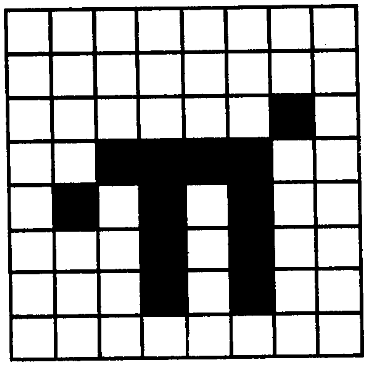
בחלקו העליון של המסך תראה מרווח, 15 סימולים וסימני פיסוק, עשר הספרות, שבעה סימולים נוספים, אותיות הדפוס, ששה סימולים נוספים, אותיות הכתיב וחמישה סימנים נוספים. כולם (מלבד £ וְ־©) נלקחו מאוסף תוים מוגדר וידוע הקרוי ASCII (ראשי תיבות של: American Standard Code for Information Interchange = קוד אמריקאי סטנדרטי לחילוף מידע). בְּ־ASCII לכל תו קיים קוד מספרי מתאים, ואלו הם הקודים בהם משתמש ZX Spectrum.
שאר התוים אינם חלק מקוד ASCII והם מיוחדים למחשב זה. הראשונים בהם הם מרווח וְ־15 תבניות בשחור־לבן. אלו הם — 'סימולים גרפיים' (graphic symbols) והשימוש בהם הוא ליצירת תמונות.
אתה יכול להדפיס סימולים אלו ישירות מלוח־המקשים, בפעולה באופן־העבודה־הגרפי (Graphic Mode). באם תקיש על GRAPHICS (CAES SHIFT עם 9) — ישתנה הסמן לְ־G. כעת מקשי הספרות מְ־1 עד 8 יהיו מקשי ה'סימולים־הגרפיים': הם ייצרו את הסימולים המשורטטים לידם; בעזרת כפתור SHIFT כלשהוא — הם מייצרים את אותם הסימולים כשהם הפוכים — דהיינו: שחור הופך ללבן ולהפך.
ללא תלות בְּ־SHIFT, מקש 9 מחזיר את המחשב לאופן הפעולה הרגיל (L) ומקש Ø הוא DELETE.
לפנינו 16 ה'סימולים הגרפים':

אופן קבלתו	הקוד	הסימול	אופן קבלתו	הקוד	הסימול
G shifted 8	143		G 8	128	
G shifted 1	142		G 1	129	
G shifted 2	141		G 2	130	
G shifted 3	140		G 3	131	
G shifted 4	139		G 4	132	
G shifted 5	138		G 5	133	
G shifted 6	137		G 6	134	
G shifted 7	136		G 7	135	

לאחר ה'סימולים־הגרפים' תראה אוסף הנראה כהעתק נוסף של האותיות מְ־A ועד U. אלו הנם תוים אותם אתה יכול להגדיר מחדש לפי־רצונך, למרות שכאשר המחשב מופעל הם מוגדרים כאותיות הללו. הם נקראים — 'גרפיקה מוגדרת — מפעיל' (user defined graphics). תוכל להדפיס תוים אלו מלוח המקשים מעבר לאופן־הפעולה־הגרפי בעזרת מקשי האותיות A עד U.

כדי להגדיר תו חדש בעצמך, עקוב אחר הדוגמא הבאה — המגדירה תו שיציג Π.

(i) בחן כיצד נראה התו אותו אתה מבקש לייצר. כל תו בנוי ממערך־רבועי של 8×8 נקודות כשכל אחת יכולה להיות ב'צבע הניר' (paper colour) או ב'צבע הדיו' (ink colour). על כך ראה גם בספרון המבוא. צייר סכימה שתתאר את התו בהתאם למערך הנקודות, כאשר נקודות ב'צבע־דיו' יסומנו בעזרת רבועים שחורים:



הותרנו מרווח ברוחב משבצת אחת בקצוות כדי להפריד מן התוים השכנים. (לא כך מתקיים באותיות־כתיב עם 'זנב' — כגון y, p ועוד — המגיע לתחתית תבנית התו).

(i i) החלט איזה סימול־גרפי ייצג את Π; הבה נחליט כי תהיה זו האות P, כך שכאשר תקיש על P כאשר המחשב נמצא באופן־הפעולה־הגרפי — ישורטט Π במסך.

(i i i) אחסן בזכרון את תבנית הצורה החדשה. כל סימול־גרפי־מוגדר־מפעיל נשמר כצרוף של 8 מספרים, אחד לכל שורה. אתה יכול לרשום כל אחד ממספרים אלו כְּ־BIN (Binary Number = מספר בינארי) שלאחריו שמונה ∅או 1 — ∅ עבור 'צבע הניר' וְ־1 עבור 'צבע הדיו'; כך שעבור הַ־Π אותו שרטטנו נקבל את 8 המספרים הבאים:

```
BIN 00000000
BIN 00000000
BIN 00000010
BIN 00111100
BIN 01010100
BIN 00010100
BIN 00010100
BIN 00000000
```

(באם אתה מכיר מספרים בינארים, יהיה זה לעזור לך לדעת כי BIN משמשת לשם כתיבת מספר בינארי — במקום מספר בבסיס עשר).

שמונה מספרים אלו נשמרים בזכרון, בשמונה מקומות, כשלכל אחד מהם כתובת. כתובתו של ה"בית" (byte) הראשון (שהוא קבוצת 8 הספרות) היא — "USR "P (P מאחר וזו היתה בחירתנו בסעיף (i i)), זו של השניה היא 1 + "USR "P וכך הלאה עד 7 + "USR "P.

USR הינה פונקציה המתרגמת ארגומנט־מחרוזת לכתובת של הבית הראשון בזכרון עבור הגרפיקה־מוגדרת־המפעיל המתאימה. ארגומנט המחרוזת חייב להיות תו יחיד אשר יכול להיות — או סימול־גרפי־מוגדר־מפעיל או האות המתאימה (דפוס או כתיב). שימוש נוסף ל־USR הוא כאשר הארגומנט הינו מספר — והוא יטופל גם־כן. גם באם־אינך מבין זאת, התכנית הבאה תבהיר זאת:

```
10 FOR n=0 TO 7
20 INPUT row: POKE USR "P"+n,row
30 NEXT n
```

התכנית תעצור 8 פעמים כדי לקבל קלט באמצעות הצהרות INPUT, כדי לאפשר לך להדפיס את שמונת מספרי הַ־BIN; הדפס אותם בסדר הנכון — החל מהשורה העליונה.
הצהרת POKE אוגרת מספר — ישירות בכתובת בזכרון המחשב תוך עקיפת המנגנון בו.משתמשת, באופן רגיל, שפת BASIC. ההפוך ל־POKE הוא PEEK והוא מאפשר לנו לבחון את תוכן הזכרון במיקום המצויין, ללא שנוי תוכן מיקום זה. הטיפל בשתי הצהרות אלו ינתן בפרק 24.
לאחר הסימולים־הגרפיים־מוגדרי־המפעיל מגיע תורם של צירופי־הסימולים (tokens).
לבטח שמת לב כי לא הדפסנו את 32 התוים הראשונים — בקודים Øעד 31. אולו הם תוי הבקרה. אין הם מייצרים תו בר־הדפסה, אולם להם השפעה מעט־ממשית על הטלויזיה או שהם משמשים לבקרת התקנים אחרים מלבד הטלויזיה, ואז הטלויזיה תציג? כדי להראות שאין היא מבינה אותם. הם מתוארים ביתר פרוט בנספח A.
התווים בהם משתמשת הטלויזיה הם אלו שלהם הקוד 6, 8, 13. ככלל CHR $8 הוא כנראה היחיד בו תמצא שמוש.
CHR $6 מדפיס מרווחים באותה דרך בדיוק כפי שעושה הפסיק — בהצהרת PRINT, לדוגמא

PRINT 1 ;CHR $ 6;2

מבצע בדיוק את שמבצע הבטוי הבא:

PRINT 1 ,2

ברור שאין זו דרך יעילה לשימוש בה. דרך הכמה יותר תהיה לאמר:

LET a$ = "1" + CHR $6 + "2"
PRINT a$

CHR$ 8 מחזיר את מיקום ההדפסה הבאה — למקום אחד לפני הנוכחי באותה שורה; נסה

PRINT "1234" ;CHR $8 ;"5"

המביאה להדפסת המספר 1235.
CHR $13 מניע את סמן מיקום ההדפסה לתחילת השורה הבאה.
כמו כן, הטלויזיה נעזרת בקודים 16 עד 23; אלו מוסברים בפרקים 15 וְ־16. כל תוי הבקרה מופיעים בנספח A.
בעזרת קודי התוים אנו יכולים להרחיב את תפיסת 'סדר האלף־בית' המכסה מחרוזות המכילות תוים כלשהם, לא רק אותיות. אם במקום האוסף הרגיל של 26 אותיות האלף־בית — נשתמש באוסף המוכתב בן 256 הסימולים, בסדר המתאים לקודיהם, אזי העקרון ישמר בדיוק כמקודם. לדוגמא, מחרוזות אלו הנן בסדר האלף־בית שלהם במחשב ZX Spectrum: (שים לב לעובדה המוזרה כי אותיות כתיב מופיעות לאחר אותיות הדפוס: לכן "a" מקומה לאחר "Z"; כמו כן — למרווחים ישנה משמעות)


```
CHR$ 3+"ZOOLOGICAL GARDENS"
CHR$ 8+"AARDVARK HUNTING"
"  AAAARGH!"
"(Parenthetical remark)"
"100"
"129.95 inc. VAT"
"AASVOGEL"
"Aardvark"
"PRINT"
"Zoo"
"[interpolation]"
"aardvark"
"aasvogel"
"zoo"
"zoology
```

לפנינו כלל, למציאת הסדר של שתי מחרוזות: תחילה השווה את התוים הראשונים; באם הם שונים — לאחד שהקוד המתאים לו הוא הקטן מבין השנים — המחרוזת המתאימה היא המוקדמת מבין המחרוזות. באם הם שוים, המשך בהשוואת התוים הבאים. באם בתהליך זה אחת מן המחרוזות "נגמרת" לפני השניה, אזי מחרוזת זו (הקצרה יותר) הוא המוקדמת; אחרת — הן חייבות להיות שוות.
היחסים = ,<>,=> ,=< ,< וכן <> משמשים גם מחרוזות כמו עבור מספריים '<' פרושו 'בא-לפני' וְ־'>' פרושו 'בא אחרי', ולכן —

"AA man" < "AARDVARK"
"AARDVARK" > "AA man"

שניהם 'נכונים'.
= > וכן = > פועלים גם הם באופן דומה על מחרוזות בדומה לפעולתם על מספרים, כך שהבטוי —

"The same string" < = "The same string"

הוא 'נכון', אולם —

"The same string" < "The same string"

'אינו־נכון'.
נסה את כל שנאמר עד כה בעזרת התכנית הבאה הקולטת 2 מחרוזות ומסדרת אותן לפי סדר.

שים לב כיצד עלינו להגדיר מחרוזת C$ בעת — שאנו ממירים את a$ וְ־b$ בשורה 2Ø.

LET a$ = b$:LET b$ = a$

לא יביא לתוצאה הדרושה.
התכנית הבאה מגדירה גרפיקה־מוגדרת־מפעיל להצגת כלי שחמט:

P — חייל (pawn)
R — צריח (rook)
N — אביר (knight) — או סוס בכנויו הפופולרי

B — רץ (bishop)
K — מלך (king)
Q — מלכה (queen)

```
5 LET b=BIN 01111100: LET c=BIN
  00111000: LET d=BIN 00010000
10 FOR n=1 TO 6: READ p$: REM 6 pieces
20 FOR f=0 TO 7: REM read piece into 8 bytes
30 READ a: POKE USR p$+f,a
40 NEXT f
50 NEXT n
100 REM bishop
110 DATA "b",0,d, BIN 00101000, BIN 01000100
120 DATA BIN 01101100,c,b,0
130 REM king
140 DATA "k",0,d,c,d
150 DATA c, BIN 01000100,c,0
160 REM rook
170 DATA "r",0, BIN 01010100,b,c
180 DATA c,b,b,0
190 REM queen
200 DATA "q",0, BIN 01010100, BIN 00101000,d
210 DATA BIN 01101100,b,b,0
220 REM pawn
230 DATA "p",0,0,d,c
240 DATA c,d,b,0
250 REM knight
260 DATA "n",0,d,c, BIN 01111000
270 DATA BIN 00011000,c,b,0
```

שים לב כי ∅נתן לשימוש במקום — BIN∅∅∅∅∅∅∅∅.
לאחר הרצת התכנית — בחן את כלי השחמט על ידי מעבר לאופן־הפעולה־הגרפי.

תרגילים:

1. דמיין לעצמך את המרווח בסימול אחד כשהוא מחולק לארבעה רבעים: אזי באם כל רבע יכול להיות שחור או לבן, ישנן 16 =,2×2×2×2 אפשרויות. מצא את כולן באוסף התוים.
2. הרץ את התכנית:

```
1∅ INPUT a
2∅ PRINT CHR $a;
3∅ GO TO 1∅
```

באם תנסה אותה, תמצא כי CHR$ מעגלת למספר הקרוב ביותר; באם a אינה בתחום שבין ∅ ל־255 התכנית עוצרת עם דווח־השגיאה: B integer out of range.
3. מבין השניים הבאים, מיהו הראשון מביניהם? "EVIL"
"evil"
4. חשוב כיצד נתן לשנות את התכנית כך שנתן יהיה להגדיר גרפיקה־מוגדרת־מפעיל כך שהיא תשתמש בהצהרות READ ו־DATA במקום הצהרת INPUT.

פרק 15: עוד אודות
PRINT וְ־INPUT

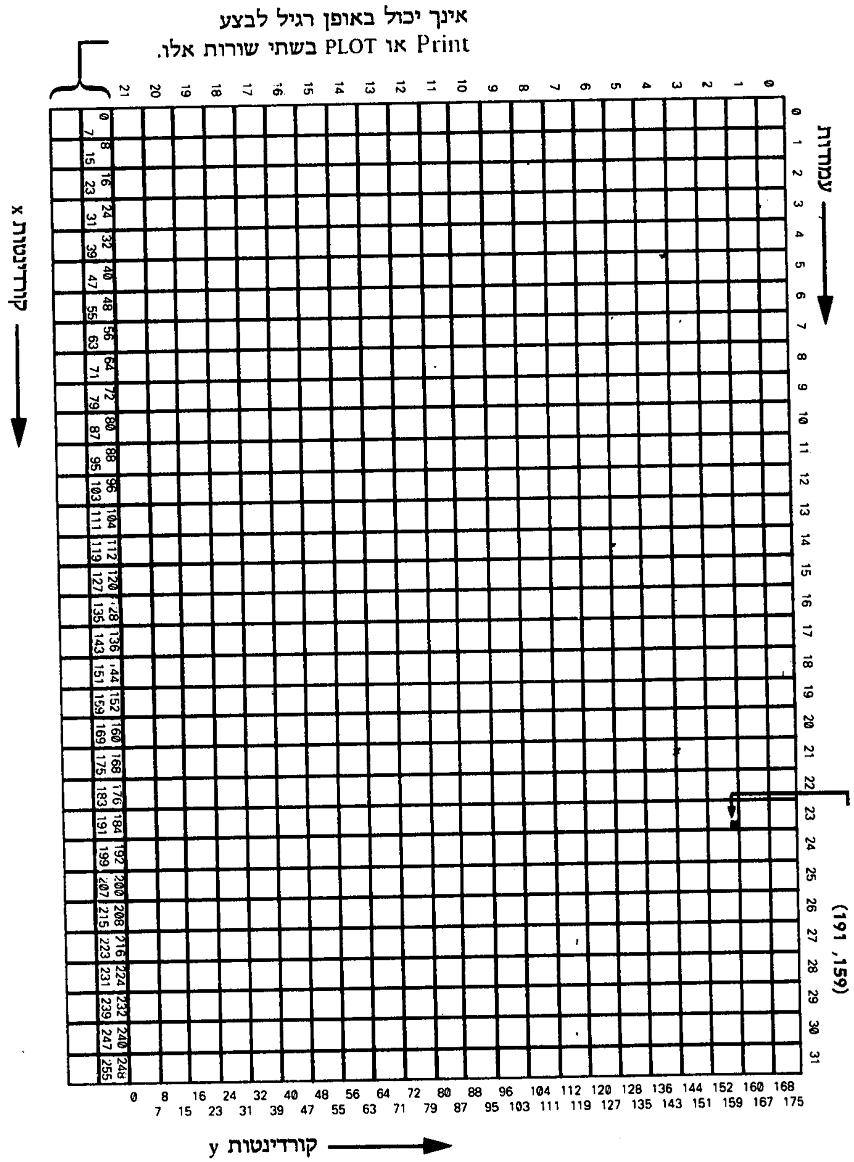
תמצית:

CLS
כלום: פרטים PRINT
בטויים (מספריים או סוגי מחרוזת): בטוי מספרי TAB, בטוי מספרי, בטוי מספרי AT
מפרידי PRINT: , ;
פרטי INPUT: משתנים (מספריים או סוגי מחרוזת)
משתנה מחרוזת LINE
כל פרטי PRINT שאינם מתחילים באות. (צרופי־סימולים [Tokens] אינם נחשבים כמתחילים עם אות)
גלגול מסך

עד כה ראית מספר פעמים רב את הצהרת PRINT — בשימוש, כך שבודאי רכשת הבנה לאופן השימוש בה. בטויים שערכיהם מודפסים באמצעות הצהרת PRINT נקראים 'פרטי־Print' (items), והם מופרדים באמצעות פסיק או פסיק־נקודה, הנקראים 'מפרידי — Print' (separators). 'פרט־Print' יכול להיות גם ריק מתוכן (דהיינו — כלום) — ובכך מוסבר המצב בו מופיעים שני פסיקים בשורה. קיימים שני 'פרטי־Print' נוספים, המשמשים להורות למחשב לא מה — אלא, היכן להדפיס. לדוגמא: PRINT AT11 ,16 ;"*" תדפיס כוכב במרכז המסך.

עמודה, שורה AT

מזיזה את מיקום ה־PRINT (המקום בו יודפס הפרט הבא) לשורה והעמודה הנקובים. השורות ממוספרות מ־∅ (למעלה) עד 21 והעמודות ∅ (משמאל) ועד 31.



עמודה TAB

מדפיסה די־מרווחים כדי להזיז את מיקום — הַPrint לעמודה המצוינת. היא תשאר באותה שורה, או באם הצהרה זו קשורה בְ־CHR $8 תעבור לשורה הבאה. שים לב — שהמחשב מפחית את מספר העמודה 'מודולו 32' (הוא מחלק ב־32 ומשתמש בשארית); לכן לְ־TAB 33 אותה המשמעות כמו לְ־TAB 1.

הדוגמא — PRINT TAB 3Ø ;1 ;"Contents" ;AT 3 ,1 ;"CHAPTER" ;TAB 24 ;"page"
מראה כיצד נתן להדפיס את כותרות דף התוכן בדף מספר 1 של ספר.
נסה להריץ את התכנית הבאה:

```
10 FOR n=0 TO 20
20 PRINT TAB 8*n;n;
30 NEXT n
```

היא מדגימה את שנאמר אודות חסור מספרי TAB — מודולו 32. כדי־להכיר דוגמא אלגנטית יותר, שנה את המספר 8 שבשורה 2Ø למספר 6.

נקודות חשובות:

(i) הדרך הטובה ביותר להפריד בין פרטים חדשים אלו היא בעזרת נקודה־פסיק, כפי שבצענו בדוגמא. אתה רשאי להשתמש בפסיקים (או — בשום־דבר בסוף הצהרה), אך משמעות הדבר היא שלאחר קביעת 'מיקום ה־PRINT' — מיד תזיז אותו בשנית, דבר שאינו יעיל כלל־ועקר.

(ii) אינך יכול להדפיס בשתי השורות התחתונות (22, 23) במסך — מאחר והן נשמרות עבור פקודות, קליטת מידע על ידי INPUT, דווחים ועוד. התיחסות ל'שורה התחתונה' כוונתה בדרך כלל לשורה 21.

(iii) אתה רשאי להשתמש בְ־AT כדי לקבוע את 'מיקום — הָ־PRINT' במקום בו קיים כבר דבר מה מודפס; כל מה שהיה־מודפס מקודם יעלם כאשר תמשיך להדפיס שם.

הצהרה נוספת הקשורה להצרת PRINT היא CLS. היא "מנקה" את כל המסך (Clear Screen) — דבר הנתן לביצוע גם בעזרת CLEAR עם RUN יחדיו.

כאשר ההדפסה מגיעה לתחתית המסך — היא מתחילה ל'גלגל' (scroll) את המסך כלפי מעלה, בדומה למכונת־כתיבה. תוכל להווכח באם תבצע

CLS :FOR n = 1 TO 22 :PRINT n :NEXT n

ולאחר מכן בצע־מספר פעמים

PRINT 99

באם המחשב מדפיס כמויות רבות של מידע, נדרשת זהירות רבה כדי להבטיח שדבר לא יגולגל מעבר לקצהו העליון של המסך בטרם הספקת לצפות בו כיאות. תוכל לראות זאת מתרחש באם תדפיס:

CLS :FOR n = 1 TO 1ØØ :PRINT n :NEXT n

לאחר שיודפס מסך מלא, יעצר התהליך תוך הדפסת השאלה — scroll? ('לגלגל'?) — בתחתית המסך. תוכל כעת לבחון את 22 המספרים הראשונים ברשימתך. כאשר תחליט לסיים עמם, הדפס y (yes-כן) והמחשב יספק לך מסך חדש מלא במספרים. למעשה, כל מקש יגרום למחשב להמשיך מלבד n (no — לא), STOP (SYMBOL SHIFT עם a) או SPACE (מקש BREAK). אלו יגרמו למחשב לחדול מהרצת התכנית — בדווח: D BREAK — CONT repeats.

הצהרת INPUT יכולה לבצע — אף יותר ממה שתארנו לפניך. ראית עד כה הצהרות INPUT מהצורה

INPUT "How old are you?" ,age

שבהן המחשב מדפיס כותרת כגון ?How old are you בתחתית המסך, ועליך להדפיס כתגובה את גילך.
למעשה, הצהרת INPUT בנויה מפרטים ומפרידים באותו־אופן בו בנויה הצהרת PRINT, ולכן How old are you וכן age מהווים שניהם פרטי INPUT. פרטי INPUT הנם בדרך כלל מאותה צורה כפרטי PRINT, אך קיימים מספר הבדלים חשובים.

ראשית, פרט נוסף בהצהרת INPUT הינו המשתנה אשר את ערכו עליך להדפיס — age בדוגמא שלנו. הכלל הוא שאם פרט — INPUT מתחיל באות הוא חייב להיות משתנה שאת ערכו יש לקבל כקלט.

שנית, המשמעות הנובעת מכך היא שאינך יכול להדפיס את ערכי — משתנים כחלק מן הכותרת; בכל אופן, תוכל לעקוף זאת על־ידי סוגריים סביב למשתנה. כל בטוי המתחיל באות חייב להיות בתוך סוגריים באם דרוש שיודפס כחלק מהכותרת.

כל פרט PRINT שהוא — שאינו מושפע מכללים אלו, הינו גם כן פרט INPUT. לפנינו דוגמא להדגים זאת:

```
LET my age = INT (RND * 100): INPUT ("I am ";my age; ". ");"How
old are you?", your age
```

my age נמצא בתוך סוגריים ולכן ערכו יודפס, your age לעומת זאת אינו בתוך סוגריים ולכן עליך להדפיס את ערכו.

כל דבר אותו רושמת הצהרת INPUT עובר לתחתית המסך, המתנהגת באורח עצמאי קמעא מהמחצית העליונה של המסך. בפרט, השורות בה ממוספרות באופן יחסי לשורה העליונה של המחצית התחתונה, גם אם הדבר הביא לתנועת המסך כלפי מעלה (כפי שקורה כאשר קיים מידע רב מאד מהצהרות INPUT).

כדי לבחון כיצד פועלת AT בהצהרות INPUT, הרץ את:

```
10 INPUT "This is line 1.",a$; AT 0,0;"This is line 0.",a$; AT 2,0;
  "This is line 2.",a$; AT 1,0;"This is still line 1.",a$
```

(לחץ על ENTER בלבד, בכל פעם שהתכנית נעצרת). כאשר מודפס — .This is line 2 — החלק התחתון של המסך נע כלפי מעלה, חלקו העליון אינו מושפע כלל — עד לשלב בו החלק התחתון מאיים לפלוש לתחומו ולכתוב בשורה המתאימה ל'מיקום ה־PRINT'. במצב זה — החלק העליון מתחיל גם הוא ל'התגלגל' כלפי מעלה — כדי למנוע זאת.

נקודה נוספת שאותה לא ראינו בהקשר להצהרת INPUT — היא LINE, שהיא דרך אחרת לקליטת משתני־מחרוזות. באם תרשום LINE לפני שמו של משתנה — מחרוזת הנדרש כקלט, כמו בדוגמא הבאה

INPUT LINE a$

אזי המחשב לא יוסיף את המרכאות אותן הוא רושם תמיד למשתנים במחרוזות, למרות שיתנהג כאילו הן קיימות שם. לכן אם תרשום

cat

כמידע — INPUT, a$ יקבל את הערך cat. מאחר ומרכאות מחרוזת אינן מופיעות במחרוזת, אינך יכול למחוק אותן ולהדפיס בטוי מחרוזת שונה כמידע — INPUT. זכור כי אינך יכול להשתמש ב־LINE עבור משתנים מספריים.

לתוי הבקרה CHR $22 ו־CHR $23 השפעות דומות לאלו של AT ו־TAB. הם מוזרים במקצת כתוי בקרה, מאחר ובעת ואחד נשלח למסך לשם הדפסה הוא חייב להיות מלווה בשני תוים נוספים שלהם

השפעה שונה מהרגיל: הם נחשבים כמספרים (הקודים שלהם) כדי להגדיר את השורה והעמודה (עבור AT) או את מיקום־התוית (עבור TAB). אתה תמצא לבטח, כי נוח יותר להשתמש ב־AT ו־TAB בדרך הרגילה מאשר בתוי הבקרה, אך יתכן ותמצא גם להם שימוש בנסיבות מסוימות. תו הבקרה עבור AT הוא CHR $22. התו הראשון שלאחריו מגדיר את מספר השורה והשני את מספר העמודה, כך שהבטוי

PRINT CHR$22 + CHR $1 + CHR$ c;

הוא בעל משמעות זהה לבטוי —

PRINT AT 1 ,c;

הדבר הינו כך — גם אם ל־CHR$ 1 או CHR$ c משמעות אחרת בדרך כלל (לדוגמא באם c = 13); CHR$ 22 לפניהם הוא המגדיר זאת.

תו הבקרה של TAB הינו CHR$ 23 ושני התוים שלאחריו משמשים להגדרת מספר בתחום ∅ עד 65535 הקובע את המספר שיהיה כפרט ה־TAB:

PRINT CHR$ 23 + CHR $a+ CHR $b;

היא בעלת אותה השפעה כמו —

PRINT TAB a + 256 * b;

אתה יכול להעזר ב־POKE כדי להורות למחשב לחדול מהשאלה — ?scroll, על־ידי:

POKE 23692 ,255

בכל עת שתחפוץ. לאחר זאת הוא יגלגל 255 פעם בטרם יעצור וישאל ?scroll. כדוגמא, נסה:

```
10 FOR n=0 TO 10000
20 PRINT n: POKE 23692,255
30 NEXT n
```

וראה כיצד נעלם הכל מן המסך!

תרגילים:

1. נסה את התכנית הבאה עם מספר ילדים, כדי לבחון את ידיעותיהם בכפל.

```
 10 LET m$=""
 20 LET a=INT (RND*12)+1: LET b=INT (RND*12)+1
 30 INPUT (m$) ' ' "what is ";(a);" * ";(b);"?";c
100 IF c=a*b THEN LET m$="Right.": GO TO 20
110 LET m$="Wrong. Try again.": GO TO 30
```

באם הם נבונים, הם עשויים לגלות שאין צורך שיבצעו את החישובים בעצמם. לדוגמא, באם המחשב מבקש מהם להדפיס את התשובה ל־3*2, כל שעליהם לעשות הוא להדפיס 3*2.
דרך אחת לעקוף זאת היא לגרום להם להכניס מחרוזות במקום מספרים. החלף את C שבשורה 3∅ ב־C$ ובשורה 1∅∅ ב־VAL C$ והכנס לתכנית שורה נוספת:

```
40 IF c$<> STR$ VAL c$ THEN LET m$="Type it properly, as
   a number.": GO TØ
```

ובכך שיטית בהם, לאחר מספר ימים, לבטח אחד מהם יגלה שהם יכולים לעקוף גם את־זאת על ידי מחיקת מרכאות המחרוזת והדפסת (2*3) STR $. כדי למנוע זאת, אתה יכול להחליף את C$ שבשורה 3Ø ב־LINE C$.

פרק 16 : צבעים

תמצית:

INK, PAPER, FLASH, BRIGHT, INVERSE, OVER,
BORDER

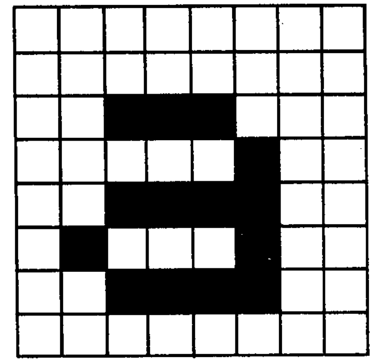
הרץ את התכנית הבאה:

```
 10 FOR m=0 TO 1: BRIGHT m
 20 FOR n=1 TO 10
 30 FOR c=0 TO 7
 40 PAPER c: PRINT "    ";: REM 4 coloured spaces
 50 NEXT c: NEXT n: NEXT m
 60 FOR m=0 TO 1: BRIGHT m: PAPER 7
 70 FOR c=0 TO 3
 80 INK c: PRINT c;"   ";
 90 NEXT c: PAPER 0
100 FOR c=4 TO 7
110 INK c: PRINT c;"   ";
120 NEXT c: NEXT m
130 PAPER 7: INK 0: BRIGHT 0
```

תכנית זו מציגה את שמונת הצבעים (כולל שחור ולבן) ושתי דרגות הבהירות אותן נתן להשיג במסך הטלויזיה — בחיבור למחשב זה. (באם הטלויזיה שברשותך היא מסוג שחור־לבן, תראה גוונים שונים של אפור). לפנינו רשימה שלהם — לשם יחוס; הם רשומים גם על המקשים המתאימים:

שחור — Ø
כחול — 1
אדום — 2
סגול — 3
ירוק — 4
תכלת — 5
צהוב — 6
לבן — 7

על מסך טלויזיה שחור־לבן, מספרים אלו מייצגים דרגות בהירות שונות.
כדי להשתמש בצבעים אלו כיאות, עליך להבין מעט יותר כיצד בנויה תמונת הצבע.
התמונה מחולקת ל־768 (24 שורות של 32) מקומות בהן יכולים תוים להיות מודפסים. כל תו מודפס כמערך של 8x8 נקודות כמו זה שלפנינו, המציג את האות a:



הדבר צריך להזכיר לך את סימולי הגרפיקה־מוגדרת־המפעיל עליהם קראת בפרק 14, שם ראינו כי Ø מייצג את הנקודות הלבנות וְ־1 את הנקודות השחורות.
מיקום התו קשור גם הוא בשני צבעים: 'הדיו' (ink) — או 'צבע החזית' — שהוא צבען של הנקודות השחורות בריבוע שלנו, וה'נייר' (paper) — או 'צבע הרקע' — שהוא צבען של הנקודות הלבנות. בהתחלה, לכל מיקום מוגדרים 'דיו' שחורה ו'נייר' לבן — כך שהרישום יופיע כשחור על גבי לבן.
מיקום התו קשור גם בדרגת בהירות (Brightness) — רגילה או בהירה במיוחד, ובאמצעי לבטא באם יתקיים הבהוב (Flashing) או לאו — כאשר ההבהוב מתקבל על ידי החלפת צבעי ה'דיו' וה'ניר' בינם לבין עצמם. כל זאת נתן לקידוד, כמספרים. אם כך — למיקום התו יוגדרו:
(i) מערך רבועי של 8×8 נקודות של Ø וְ־1 להגדרת תבנית התו, כאשר Ø הוא עבור 'נייר' וְ־1 עבור 'דיו'.
(ii) צבעי 'ניר' ו'דיו' כשכל אחד מהם מתורגם למספר בין Ø לְ־7.
(iii) בהירות — Ø עבור רגילה וְ־1 לבהירות גבוהה.
(iv) הבהוב. — Ø לחוסר־הבהוב וְ־1 ליצירת הבהוב.
שים לב שמאחר וצבעי ה'ניר' וה'דיו' מכסים מיקום תו — שלם, אינך יכול שיהיו יותר משני צבעים במערך של 64 נקודות. אותו מצב מתקיים גם לגבי הבהירות וההבהוב: הם מתייחסים לכל מיקום התו ולא לנקודות מסוימות במערך הנקודות.
הצבעים, הבהירות ומספר ההבהוב בכל מיקום שהוא נקראים 'תכונות' (attributes).
בעת שאתה מדפיס משהוא על המסך — אתה משנה את תבנית הנקודות באותו מיקום; הדבר ברור פחות, אך עדיין נכון, כי בכך אתה משנה את התכונות באותו מיקום.
מיד־לאחר הפעלת המחשב, אינך צריך לשים לבך — לכך, מאחר והכל יודפס ב'דיו' שחורה על 'ניר' לבן (בבהירות רגילה וללא הבהוב), אך אתה יכול לשנות זאת בעזרת ההצהרות INK, PAPER, BRIGHT וְ־FLASH.
נסה —

PAPER 5

והדפס תוים נוספים כלשהם; הם יופיעו כולם על רקע תכלת, מאחר ובעת שהם מודפסים — צבע ה'ניר' במיקומים בהם יודפסו התוים — הופך לתכלת (המתאים לקוד 5).
האחרים פועלים באופן דומה, כך שלאחר —

PAPER מספר בין Ø לְ־7
INK מספר בין Ø לְ־7
BRIGHT Ø או 1
או:
FLASH Ø או 1

חשוב על Ø כְּ־off (לא פעיל) ועל 1 כעל on (פעיל)

כל הדפסה מגדירה את התכונות המתאימות בכל מיקומי־התוים בהן היא מדפיסה. נסה כמה מאלה — אתה אמור להיות כעת בעל־יכולת לבחון כיצד התכנית בתחילת הפרק פעלה (זכור כי מרווח הינו תו שבו INK וְ־PAPER הם באותו צבע.)
קיימים מספרים נוספים בהם תוכל להשתמש בהצהרות אלו שלהם השפעה־ישירה קטנה יותר.
8 נתן לשימוש בכל אחת מ־4 ההצהרות הללו במשמעות 'שקוף' — במובן שהתכונה הקודמת מראה־מספיק. נניח־לדוגמא כי בצעת: PAPER 8
אף מיקום־תו לא יהיה בצבע 'ניר' בערך 8 מאחר ולא קיים צבע שכזה; מה שיקרה הוא שכאשר מיקום זה יודפס — צבעו ישאר ללא שנוי — כפי שהיה בפעם הקודמת. INK 8, BRIGHT 8 וכן FLASH 8 פועלים באופן דומה, על התכונות האחרות.
9 נתן לשימוש רק עבור PAPER וְ־INK ופרושו: 'יצירת ניגודי־גוון'. הצבע ('דיו' או 'ניר') בו אתה משתמש, מושווה עם האחר ובעקבות זאת — הוא הופך ללבן באם השני הינו כהה (שחור, כחול, אדום או סגול), ולשחור באם השני הינו צבע בהיר (ירוק, תכלת, צהוב, או לבן). נסה זאת ע"י הדפסת

INK 9 :FOR C = Ø TO 7 :PAPER C :PRINT C :NEXT n

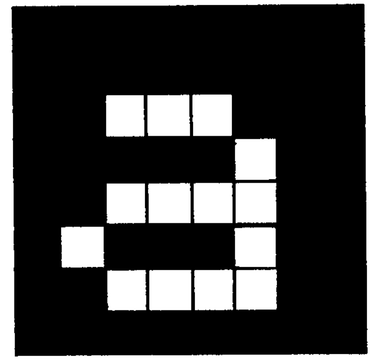
תצוגה מרשימה יותר של עוצמת הדברים היא להריץ את התכנית שבתחילת הפרק — ליצירת רצועות צבע, ולאחר מכן לבצע:

INK 9 :PAPER 8 :PRINT AT Ø ,Ø ; :FOR n = 1 TO 1ØØØ :PRINT n; : NEXT n

צבע ה'דיו' כאן מוגדר כדי ליצור 'נגודי־גוון' עם צבע ה'ניר' הקודם בכל מיקום.
טלויזיות צבעוניות נשענות על העובדה המוזרה קמעא שעון האדם יכולה למעשה להבחין רק בין שלושה צבעים — בין שלושת הצבעים הראשוניים שהם: כחול אדום וירוק. כל יתר הצבעים הנם תוצאה של ערבוב שלושת הצבעים הראשוניים. לדוגמא — סגול מתקבל מערבוב אדום עם כחול, וזו גם הסיבה מדוע הקוד המתאים לו הוא 3 — שהוא סכום קודי הצבעים כחול ואדום.
כדי לראות כיצד כל שמונת הצבעים מתאימים יחדיו, דמיין לעצמך 3 נקודות־אור מלבניות בצבעים כחול, אדום וירוק המבהיקות לא בדיוק באותו מיקום על פיסת נייר במקום חשוך. כאשר הן חופפות תראה תערובת של צבעים, כפי שמציגה התכנית הבאה: (שים לב שמרווחי 'דיו' מושגים ע"י SHIFT עם 8 באופן־הפעולה־הגרפי).

```
  10 BORDER 0: PAPER 0: INK 7: CLS
  20 FOR a=1 TO 6
  30 PRINT TAB 6; INK 1;"██████████████████": REM 18 ink
     squares
  40 NEXT a
  50 LET dataline=200
  60 GO SUB 1000
  70 LET dataline=210
  80 GO SUB 1000
  90 STOP
 200 DATA 2,3,7,5,4
 210 DATA 2,2,6,4,4
1000 FOR a=1 TO 6
1010 RESTORE dataline
1020 FOR b=1 TO 5
1030 READ c: PRINT INK c;"██████";: REM 6 ink squares
1040 NEXT b: PRINT : NEXT a
1050 RETURN
```

קיימת פונקציה בשם ATTR המוצאת את התכונות הקיימות במיקום מוגדר במסך. זוהי פונקציה מסובכת מעט ולכן נטפל בה בהמשך הפרק.
קיימות שתי הצהרות נוספות — INVERSE וְ־OVER השולטות לא על התכונות, כי אם על תבנית הנקודות המודפסת במסך. הן משתמשות במספרים Ø וְ־1 עבור off וְ־on בהתאמה (לא־פעיל ופעיל) בדומה להצהרות BRIGHT וְ־FLASH. אולם אלו הן האפשרויות היחידות. באם תבצע INVERSE 1 — תקבל תבנית תו הפוכה מצורתה הרגילה: נקודות 'ניר' יוחלפו בנקודות 'דיו' ולהפך. לכן הַ־ a שהודפסה קודם — תראה כעת כך:



באם ברגע ההפעלה מתקיימים — 'דיו' שחורה ו'ניר' לבן — אזי a זה יופיע כלבן על רקע שחור — אך זכור כי גם עבור תו זה ה'דיו' הינה 'שחורה' וה'ניר' לבן! הנקודות הן אלו אשר השתנו.
ההצהרה

OVER 1

מפעילה צורת הדפסה — חוזרת מיוחדת. באופן רגיל כאשר משהוא מודפס במיקום תו כלשהוא — הוא מוחק לחלוטין את התו שהיה שם קודם; אולם בעת התו החדש יתווסף פשוט על הקודם (אולם — ראה תרגיל 1). הדבר יכול להיות שמושי מאד לשם רישום תוים מורכבים — כגון אותיות עם הדגשות עליהן, כמו בתכנית הבאה המדפיסה מאותיות השפה הגרמנית — 'ס' עם תו היגוי מעליה. (בצע NEW תחילה).

```
10 OVER 1
20 FOR n=1 TO 32
30 PRINT "o"; CHR$ 8;"""";
40 NEXT n
```

(שים לב לתו הבקרה CHR $8 המחזיר מיקום אחד חזרה).
קיימת דרך יעילה יותר לשימוש בְּ־INK וְ־PAPER מאשר השימוש בהן כהצהרות; תוכל להגדירן כפרטים בהצהרת PRINT (כשלאחריהן הסימן ;) ואז הן יבצעו בדיוק.אותו דבר כאילו עמדו כהצהרות בפני עצמן מלבד העובדה שהשפעתן היא זמנית: היא נמשכת כל עוד לא תמה פקודת הַ־PRINT בה הן מוגדרות. לכן אם תדפיס:

PRINT PAPER 6; "X";:PRINT "Y"

אזי, רק X יודפס בצהוב.
INK והשאר, כאשר מתקיימות כהצהרות בפני עצמן — אינן משפיעות על הצבעים של החלק התחתון של המסך, שם מודפסות פקודות ומידע — INPUT. החלק התחתון משתמש בצבע הגבול כצבע ה'ניר' וקוד 9 עבור 'ניגוד הגוון' עבור צבע ה'דיו', ללא הבהוב ובבהירות רגילה. תוכל לשנות את צבע הגבול לכל אחד משמונת הצבעים (לא 8 או 9) בהעזר בהצהרה:

BORDER צבע

כאשר תדפיס מידע — INPUT, גם הוא נשמע לכלל של צבע 'דיו' הנוגד בגוונו לצבע הגבול; אולם אתה יכול לשנות את צבע הכותרות המודפסות על ידי המחשב בעזרת פרטי INK וְ־PAPER (וכך הלאה) בהצהרת הַ־INPUT, כפי שהיית מבצע בהצהרת PRINT. השפעתם היא עד לתום ההצהרה או עד להדפסת מידע — INPUT, בהתאם למתרחש קודם. נסה:

INPUT FLASH 1 ;INK 1 ;"What is your namber?" ;n

קיימת דרך נוספת לשנוי הצבעים — בעזרת תוי־הבקרה, בדומה לתוי הבקרה עבור AT וְ־TAB שבפרק 15.

INK	מתאימה להצהרה	CHR$ 16
PAPER	מתאימה להצהרה	CHR$ 17
FLASH	מתאימה להצהרה	CHR$ 18
BRIGHT	מתאימה להצהרה	CHR$ 19
INVERSE	מתאימה להצהרה	CHR$ 20
OVER	מתאימה להצהרה	CHR$ 21

לכל אלו מתלווה תו המייצג את הצבע בהתאם לקוד: לכן (כדוגמא)

PRINT CHR $16 + CHR $9 ;...

הוא בעל אותה משמעות כמו —

PRINT INK 9;...

בכלל, לא תטרח להשתמש בתוי בקרה אלו, מאחר ותוכל להשתמש בפרטי הצבעים. אולם, דבר יעיל שתוכל לבצע באמצעותם הוא להכניסם לתכניות: הדבר מאפשר שחלקים שונים יופיעו ברישום התכנית (listing) בצבעים שונים, כדי להבדיל אותם זה מזה או אפילו רק כדי שהתכנית תראה יפה. אתה חייב לרשום אותם לאחר מספר השורה, אחרת הם פשוט יאבדו.
כדי להכניסם לתכנית, עליך להכניסם מלוח המקשים, בדרך־כלל בהעזר באופן הפעולה המורחב (E) עם הספרות.
הספרות Ø עד 7 מגדירות את הצבעים המתאימים — ל'דיו' באם CAPS SHIFT גם הוא נלחץ ול'ניר' באם לאו. ליתר דיוק, באם אתה נמצא באופן־הפעולה־המורחב (E) ואתה מקיש על ספרה כלשהיא (הבה נניח כי 6 — עבור צהוב; בכל מקרה — ערך זה חייב להיות בין Ø לְ־7 — לא 8 או 9) אזי מוכנסים שני תוים: הראשון $ CHR שמשמעותו PAPER וְ־CHR $6 שמשמעותו — 'בחר צבע צהוב'. באם לחצת על CAPS SHIFT באותו זמן, תקבל CHR $16 שמשמעותו — 'הגדר את צבע הדיו' במקום — CHR $17
מאחר ואילו שני תוים, אתה יכול לקבל תופעות מוזרות: בעת שתמחוק אותם — עליך להקיש על DELETE פעמיים, ולאחר הפעם — הראשונה יתקבל לעיתים קרובות סימן שאלה או אפילו סימנים מוזרים יותר. אל תדאג; רק — הקש על DELETE פעם נוספת.

→ ו ← יכולים גם הם להתנהג בצורה מוזרה בעת שהסמן חולף על פני תוי־הבקרה.
אך, עדיין באופן־הפעולה־המורחב (E).
8 מביא ל־CHR $19 ו־CHR Ø ´ לבהירות רגילה
9 מביא ל־CHR $12 ו־CHR $1 לבהירות גבוהה.
CAPS SHIFT עם 8 מביא ל־CHR$18 ו־CHRØ לחוסר הבהוב
CAPS SHIFT עם 9 מביא ל־CHR$19 ו־CHR 1 עבור הבהוב

קיימים כמה נוספים באופן הפעולה הרגיל (L):

CAPS SHIFT עם 3 מביא ל־CHR $20 ו־CHR$ Ø לתוים רגילים
CAPS SHIFT עם 4 מביא ל־CHR $20 ו־CHR $1 לתוים הפוכים.

לסיכום האמור לעיל, הבה נבחן את הטבלה הבאה — המתארת את השורה העליונה בלוח המקשים:

MODE	SHIFT										
E	סימבול	DEF FN	FN	LINE	OPEN #	CLOSE #	MOVE	ERASE	POINT	CAT	FORMAT
	CAPS	דיו כחולה	דיו אדומה	דיו סגולה	דיו ירוקה	דיו תכלת	דיו צהובה	דיו לבנה	ללא הבהוב	עם הבהוב	דיו שחורה
	אף אחד	נייר כחול	נייר אדום	נייר סגול	נייר ירוק	נייר תכלת	נייר צהוב	נייר לבן	בהירות רגילה	בהירות גבוהה	נייר שחור
G	אחד משניהם	▙	▟	▄	▛	▌	▚	▖	█	EXIT GRAPHICS	DELETE
	אף אחד	▝	▘	▀	▗	▐	▞	▜		EXIT GRAPHICS	DELETE
K,L or C	CAPS	EDIT	CAPS LOCK	TRUE VIDEO	INVERSE VIDEO	⇦	⇩	⇧	⇨	GRAPHICS MODE	DELETE
	SYMBOL	!	@	#	$	%	&	'	(	)	_
	אף אחד	1	2	3	4	5	6	7	8	9	Ø

לפונקציה ATTR הצורה הבאה —

ATTR (עמודה, שורה)

שני הארגומנטים שלה הם מספרי השורה והעמודה בהם תשתמש כפרטי הצהרת AT, והתוצאה המתקבלת היא מספר המראה את התכונות (הצבע וכו') במיקום המתאים לארגומנטים — במסך. אתה יכול להשתמש בה בחופשיות בבטויים כפי שאתה משתמש עם פונקציה כלשהיא.
המספר המתקבל כתוצאה הוא סכום ארבעת המספרים האחרים כדלהלן:

128 באם מיקום התו — מהבהב, Ø באם הוא יציב
64 באם מיקום התו בהיר במיוחד, Ø באם הוא רגיל
8* הקוד המתאים לצבע ה'ניר'.
קוד צבע ה'דיו'.

לדוגמא, באם מיקום התו מהבהב, בהירותו רגילה וצבע ה'ניר' בו ירוק עם 'דיו' כחולה, אזי ארבעת המספרים שעלינו לחבר יחד הם — 128, Ø, 48 = 8*6 וכן 1 שסכומם יחד נותן 177. בחן זאת ע"י: PRINT AT Ø,Ø ;FLASH 1;PAPER 6 ;INK 1 ;""; ATTR (Ø,Ø)

תרגילים:

1. נסה את —

```
PRINT "B" ;CHR$ 8 ;OVER 1 ;"/";
```

כאשר, במקום בו הסימול / חוצה את B, הוא הותיר נקודה לבנה, זוהי הדרך בה הדפסה־חוזרת פועלת בְּ־ZX Spectrum: שני סוגי־'ניר' או שני סוגי 'דיו' מגדירים 'ניר', כאשר אחד מהם מגדיר 'דיו'. לכך תכונה מענינת — שאם תדפיס הדפסה־חוזרת אותו דבר בדיוק פעמיים, תקבל חזרה מה שהיה מלכתחילה.
באם תדפיס כעת

```
PRINT CHR$ 8 ;OVER 1 ;"/"
```

מדוע מתקבל B ללא כל פגם?

2. הדפס —

```
PAPER Ø :INK Ø
```

האין זה כאילו אין הם־משפיעים על החלק התחתון של המסך?
כעת הדפס —

```
BORDER Ø
```

וראה כיצד המחשב שומר את צעדיך!

3. הרץ את התכנית הבאה:

```
1Ø POKE 22527+RND*7Ø4 ,RND*127
2Ø GO TO 1Ø
```

ללא חשיבות כיצד תכנית זו פועלת, היא משנה את צבעי הרבועים על פני המסך — כאשר ה־RND מבטיחים שהדבר יקרה באופן אקראי. רצועות האלכסונים בהן אתה מבחין — הנן הצגת התבנית הנסתרת בְּ־RND, התבנית הגורמת לה להיות פסאודו אקראית — במקום אקראית לחלוטין.

4. הדפס או העזר בְּ־LOAD כדי להכניס למחשב את כלי השחמט מפרק 14, ואחר הדפס תכנית זו המציירת לוח־שחמט העזר בהם.


```
  5 REM draw blank board
 10 LET bb=1: LET bw=2: REM red and blue for board
 15 PAPER bw: INK bb: CLS
 20 PLOT 79,128: REM border
 30 DRAW 65,0: DRAW 0,-65
 40 DRAW -65,0: DRAW 0,65
 50 PAPER bb
 60 REM board
 70 FOR n=0 TO 3: FOR m=0 TO 3
 80 PRINT AT 6+2*n, 11+2*m;"  "
 90 PRINT AT 7+2*n, 10+2*m;"  "
100 NEXT m: NEXT n
110 PAPER 8
120 LET pw=6: LET pb=5: REM colours of white and black pieces
200 DIM b$(8,8): REM positions of pieces
205 REM set up initial positions
210 LET b$(1)="rnbqkbnr"
220 LET b$(2)="pppppppp"
230 LET b$(7)="PPPPPPPP"
240 LET b$(8)="RNBQKBNR"
300 REM display board
310 FOR n=1 TO 8: FOR m=1 TO 8
320 LET bc=CODE b$(n,m): INK pw
325 IF bc=CODE" " THEN GO TO 350: REM space
330 IF bc>CODE "Z" THEN INK pb: LET bc=bc-32: REM lower
case for black
340 LET bc=bc+79: REM convert to graphics
350 PRINT AT 5+n, 9+m; CHR$ bc
360 NEXT m: NEXT n
400 PAPER 7: INK 0
```

פרק 17: גרפיקה

תמצית:

CIRCLE ,DRAW ,PLOT
POINT
pixels ('מרכיבי־תמונה')

בפרק זה נראה כיצד לצייר תמונות בְּZX Spectrum. חלק המסך בו אתה יכול להשתמש בנוי מִ־22 שורות וְ32 עמודות, היוצרים יחד 704=22*32 מיקומי־תוים. כפי שאתה עשוי לזכור, מפרק 16, כל אחד ממיקומי התוים בנוי ממערך רבועי של 8×8 נקודות, ואלו קרויים pixels (Picture-elements = מרכיבי תמונה).

מרכיב־תמונה מוגדר בעזרת־שני מספרים — הקואורדינטות שלו. הראשונה, קואורדינטת ציר x — מגדירה מהו מרחקו מהעמודה השמאלית ביותר. השניה, קואורדינטת ציר y מגדירה מהו המרחק מן התחתית. קואורדינטות אלו נכתבות בדרך כלל נצמד בתוך סוגרים, ולכן (Ø,Ø), (255, Ø), (Ø ,175) וְ־(255 ,175) מגדירות את הפִינות השמאלית־התחתונה, הימנית־התחתונה, השמאלית־העליונה והימנית־העליונה בהתאמה.

ההצהרה

PLOT x קואורדינטת, y קואורדינטת

מסמנת ב'דיו' את 'מרכיב־התמונה' שבקואורדינטות הללו, כך שהתכנית הבאה —

```
1Ø PLOT INT (RND*256) ,INT (RND*176) :INPUT a$ :GO TO 1Ø
```

משרטטת נקודה אקראית בכל פעם שתלחץ על ENTER.

לפנינו תכנית מענינת יותר. היא משרטטת את הגרף של הפונקציה SIN (גל סינוסי) עבור ערכים בתחום Ø עד 2π.

```
10 FOR n=0 TO 255
20 PLOT n,88+80*SIN (n/128*PI)
30 NEXT n
```

התכנית הבאה מציירת את הגרף המתאים לפונקציה SQR (חלק מפרבולה) — עבור תחום הערכים Ø עד 4:

```
10 FOR n=0 TO 255
20 PLOT n,80*SQR (n/64)
30 NEXT n
```

שים לב כי קואורדינטות 'מרכיבי־התמונה' שונים במקצת ממספרי השורה והעמודה בפרטי AT. אתה עשוי למצוא את הסכימה בפרק 15 שמושית כדי למצוא קואורדינטות ל'מרכיבי תמונה' ולמספרי שורה ועמודה.

כדי לעזור לך בציוריך, המחשב מצייר קוים ישרים, מעגלים וכן חלקי מעגלים עבורך, בעזרת ההצהרות DRAW וְ־CIRCLE.

ההצהרה DRAW המציירת קו ישר היא מהצורה —

DRAW x ,y

מקום ההתחלה של קו זה היא המקום האחרון בו נעצרה ההצהרה PLOT, DRAW או CIRCLE (הוא נקרא מיקום ה־PLOT; RUN, CLEAR, CLS וכן NEW מאפסים אותו לפינה השמאלית־התחתונה (Ø,Ø), ומקום הסיום הוא x 'מרכיבי־תמונה' ימינה לו וְ־y מעליו. הצהרת DRAW לכשעצמה מגדירה את אורכו וכוונו של הקו אך לא את נקודת התחלתו.

נסה את ההצהרות PLOT וְ־DRAW בעזרת:

PLOTØ,1ØØ :DRAW 8Ø ,—35
PLOT 9Ø ,15Ø :DRAW 8Ø ,—35

שים לב כי המספר בְּ־DRAW יכולים להיות גם שליליים, אך ב־PLOT לא.

אתה יכול להשתמש בהצהרות אלו בצרוף עם צבע, למרות שעליך לזכור תמיד כי הצבעים מוגדרים לתו־שלם (מערך 8×8 נקודות) בעוד הקואורדינטות הללו מוגדרות ל'מרכיבי־תמונה'. בעת שמשורטט 'מרכיב־תמונה', הוא נערך להציג את צבע ה'דיו' כולו וכל מיקום התו המכיל אותו יקבל גם הוא את צבע ה'דיו' הנוכחי. התכנית הבאה מדגימה זאת:

```
10 BORDER 0: PAPER 0: INK 7: CLS : REM black out screen
20 LET x1=0: LET y1=0: REM start of line
30 LET c=1: REM for ink colour, starting blue
40 LET x2=INT (RND*256): LET y2=INT (RND*176): REM
   random finish of line
50 DRAW INK c;x2-x1,y2-y1
60 LET x1=x2: LET y1=y2: REM next line starts where last one
   finished
70 LET c=c+1: IF c=8 THEN LET c=1: REM new colour
80 GO TO 40
```

דומה שהקוים הולכים ומתרחבים ככל שהתכנית נמשכת, וזאת מאחר וקו משנה את צבעי 'מרכיבי התמונה' המסומנים בצבע 'דיו' בכל מיקומי התוים דרכם הוא חולף. שים לב כי אתה יכול לשבץ פרטי PAPER, INK, BRIGHT, FLASH, INVERSE וכן OVER בהצהרות PLOT וְ־DRAW כפי שיכולת לעשות כך בהצהרות PRINT וְ־INPUT. הן פועלות בין מילות־המקש והקואורדינטות ומסתיימות בפסיק או בפסיק־נקודה.

נקודה נוספת לגבי DRAW היא שאתה יכול להשתמש בה כדי לשרטט חלקי מעגל במקום קוים ישרים, **על ידי הוספת מספר להגדרת הזוית בה יש להסתובב: הצורה היא** —

DRAW x ,y ,a

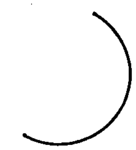
x וְ־y **משמשים להגדרת** מקום הסיום של הקו כמקודם, וְ־a היא מספר הרדיאנים אותם יש להסתובב — באם a חיובית הסיבוב הוא לכוון שמאל ובאם a שלילית — לכוון ימין. דרך אחרת לראות את a **היא** כמספר המייצג את חלק המעגל מן המעגל השלם אשר מצויר: מעגל שלם הוא 2π רדיאנים **ולכן אם** a=π — יצויר חצי־מעגל, באם a = Ø.5*π — רבע מעגל וכך הלאה.

לדוגמא, נניח כי a=π. עבור ערכי x, y כלשהם יצויר חצי־מעגל. הרץ את השורה —

```
10 PLOT 100,100: DRAW 50,50, PI
```

אשר יצייר את הצורה הבאה —

סיום ב־(15∅ ,15∅)



התחלה ב־ (1∅∅ ,1∅∅)

השרטוט יתחיל במגמה דרומית־מזרחית, אך עם הזמן היא תלך ותשתנה למגמה צפון־מערבית: ביניהם הוא הסתובב סיבוב של 18∅ מעלות, או π רדיאנים (ערכו של a).
הרץ את התכנית מספר פעמים, כאשר PI מוחלף בבטויים שונים כגון PI—, PI/2, 3*PI/2, PI/4, 1, ∅.
ההצהרה האחרונה בפרק זה היא ההצהרה CIRCLE, המציירת מעגל שלם. עליך לציין את קואורדינטות המרכז ואת הרדיוס בצורה הבאה:

CIRCLE x קואורדינטת, y קואורדינטת, רדיוס

בדומה להצהרות PLOT ו־DRAW, אתה יכול לקבוע פרטי צבע שונים בתחילת הצהרת CIRCLE —.
הפונקציה POINT מציינת באם 'מרכיב־תמונה' הוא בצבע 'דיו' או בצבע־ה'ניר'. היא בעלת שני ארגומנטים, הקואורדינטות (החייבות להופיע בסוגרים); תוצאתה היא ∅ באם הצבע הוא צבע 'ניר' ו־1 באם הוא צבע 'הדיו'. נסה:

CLS :PRINT POINT (∅ ,∅) :PLOT (∅ ,∅) :PRINT POINT (∅,∅)

הדפס —

PAPER 7 :INK ∅

והבה ונחקור כעת כיצד INVERSE ו־OVER פועלים בתוך הצהרת PLOT. שנים אלו משפיעים רק על 'מרכיב־התמונה' הרלונטי, ולא על שאר מיקומי־התוים. בדרך כלל הם ∅ (off — לא פעילים) בהצהרת PLOT, כך שעליך להזכירם רק בעת שברצונך להפעילם (1).
להלן רשימת האפשרויות לשם יחוס:
PLOT; — זוהי הצורה הרגילה. היא מביאה לציור נקודת 'דיו' — דהיינו גורמת ל'מרכיב־התמונה' להראות את צבע ה'דיו'.
PLOT INVERSE 1; — זו מציירת נקודה בצבע ההפוך ל'דיו' דהיינו צבע ה'ניר'.
PLOT OVER 1; — זו משנה את 'מרכיב־התמונה' להפוך משהיה קודם; לכן אם הוא היה צבע 'דיו' הוא יהיה כעת צבע 'ניר', ולהפך.
PLOT INVERSE 1 ;OVER 1; — זו משאירה את 'מרכיב־התמונה' בדיוק כפי שהיה קודם; אולם שים לב כי היא משנה את מיקום ה־PLOT, כך שאתה יכול בפשטות להשתמש בה למטרה זו.
כדוגמא נוספת לשימוש בהצהרת OVER, מלא את המסך בתוים הכתובים בשחור־לבן ואחר הדפס:

PLOT∅,∅ :DRAW OVER 1 ;255 ,175

שורה זו תצייר קו ישר למדי, למרות שהוא מכיל מרווחים בכל מקום בו הוא נתקל בתו — כתוב. כעת בצע שוב את אותה פקודה. הקו יעלם ללא עקבות. זהו היתרון הכביר הטמון ב־1 OVER.
באם ציירת את הקו בהתאם לשורה —

PLOT ∅,∅ :DRAW 255 ,175

ומחקת אותו בעזרת —

PLOT ∅,∅ :DRAW INVERSE 1 ;255 ,175

אזי מחקת חלק מן התוים המודפסים. כעת נסה:

PLOT ∅ ,∅: DRAW OVER 1 ;25∅ ,175

נסה למחוק אותו בעזרת

DRAW OVER 1 ;—25∅ ,—175

התהליך אינו מתבצע כמתוכנן, מאחר ו'מרכיבי־התמונה' בהם משתמש הקו בדרכו חזרה אינם בדיוק אלו שהיו בדרך הראשונה. לכן אתה חייב למחוק קו בדיוק באותו כוון בו ציירת אותו.
דרך אחת להשיג צבעים יוצאי־דופן היא לערבב שני רגילים ברבע יחיד, בעזרת גרפיקה — מוגדרת נסה תכנית זו:

```
1000 FOR n=0 TO 6 STEP 2
1010 POKE USR "a"+n, BIN 01010101: POKE USR "a"+n+1,
     BIN 10101010
1020 NEXT n
```

הנותנת את הגרפיקה־מוגדרת־המפעיל המתאימה לתבנית לוח שחמט. באם תדפיס תו זה (מעבר לאופן־הפעולה־הגרפי ולאחר מכן a ב'דיו' אדומה על 'ניר' צהוב, תמצא כי התקבל צבע כתום.

תרגילים:

1. שחק עם פרטי PAPER, INK ו־BRIGHT בהצהרת PLOT, אלו הם החלקים המשפיעים על כל מיקום־התו המכיל את 'מרכיב־התמונה'. באופן רגיל הדבר הוא כאילו הצהרת PLOT התחילה בצורה —

PLOT PAPER 8; FLASH 8; BRIGHT 8; ...

ורק צבע ה'דיו' של מיקום התו משתנה כאשר משהוא משורטט שם, אך תוכל לשנות זאת באם כך רצונך.
היה זהיר במיוחד בעת השימוש בצבעים עם INVERSE 1, מאחר וזו גורמת ל'מרכיב התמונה' להיות בצבע ה'ניר' אך משנה את צבע ה'דיו' והתוצאה עלולה להיות לא מה שציפית שתהיה.
2. נסה לצייר מעגלים בעזרת SIN ו־COS (באם קראת את פרק 10, נסה לבחון כיצד). הרץ התכנית:

```
10 FOR n=0 TO 2*PI STEP PI /180
20 PLOT 100+80*COS n,87+80*SIN n
30 NEXT n
40 CIRCLE 150,87,80
```

אתה יכול להווכח בכך שהצהרת CIRCLE היא מהירה יותר, גם אם פחות מדויקת.
3. נסה —

CIRCLE 1∅∅ ,87 ,8∅ :DRAW 5∅ ,5∅

אתה יכול ללמוד מכאן כי הצהרת CIRCLE מותירה את מיקום ה־PLOT במקום בלתי־נתן לידיעה מראש; הוא נמצא במקום כלשהוא במחצית הדרך למעלה במחצית הימנית של המעגל. תזדקק בדרך־כלל לעקוב אחר הצהרת CIRCLE עם הצהרת PLOT בטרם תמשיך לצייר עוד.
4. לפנינו תכנית המציירת גרף של כמעט כל פונקציה שהיא. היא שואלת תחילה למספר n; היא תצייר את הפונקציה עבור ערכיה שבתחום n— עד n+. לאחר־מכן היא שואלת מהי הפונקציה עצמה, וקולטת אותה כמחרוזת. המחרוזת צריכה להיות בטוי המשתמש ב־x כארגומנט הפונקציה.

```
10 PLOT 0,87: DRAW 255,0
20 PLOT 127,0: DRAW 0,175
30 INPUT s,e$
35 LET t=0
40 FOR f=0 TO 255
50 LET x=(f-128)*s/128: LET y=VAL e$
60 IF ABS y>87 THEN LET t=0: GO TO 100
70 IF NOT t THEN PLOT f,y+88: LET t=1: GO TO 100
80 DRAW 1,y-old y
100 LET old y=INT (y+.5)
110 NEXT f
```

הרץ אותה וכדוגמא הדפס 1∅ כערך עבור n ואת הבטוי 10*TAN x עבור הפונקציה. היא תצייר גרף של tanx — עבור a בתחום 10 — עד 10+.

פרק 18: תנועה

תמצית:

PEEK ,INKEY $,PAUSE

לעתים קרובות תרצה שתכנית מסוימת תפעל במשך זמן קצוב מראש, ולשם כך תמצא את ההצהרה PAUSE יעילה מאד.

PAUSE n

עוצרת את תהליך הביצוע ומציגה את התמונה למשך n מסגרות זמן של הטלויזיה (באירופה — קיימות 50 מסגרות בשניה, ובארה"ב 60). ערך n המקסימלי יכול להיות 65535, המאפשר מרווח זמן של עד 22 דקות; באם ∅ = n — משמעותו: 'PAUSE לעולמי עד'.

PAUSE נתן לקיצור על ידי לחיצה על מקש (שים לב כי — מרווח עם CAPS SHIFT מביא להפסקת התכנית גם כן). עליך ללחוץ על המקש לאחר ש־PAUSE החל את פעולתו.

התכנית הבאה מציגה את תנועת מחוגו־המשני של השעון:

```
 10 REM First we draw the clock face
 20 FOR n=1 TO 12
 30 PRINT AT 10-10*COS (n/6*PI),16+10*SIN (n/6*PI);n
 40 NEXT n
 50 REM Now we start the clock
 60 FOR t=0 TO 200000: REM t is the time in seconds
 70 LET a=t/30*PI : REM a is the angle of the second hand in
    radians
 80 LET sx=80*SIN a: LET sy=80*COS a
200 PLOT 128,88: DRAW OVER 1;sx,sy: REM draw second hand
210 PAUSE 42
220 PLOT 128,88: DRAW OVER 1;sx,sy: REM erase second hand
400 NEXT t
```

השעון יפעל בערך עד 55.5 שעות עקב שורה 6∅, אולם אתה יכול בפשטות להאריך את משך פעולתו. שים לב כיצד התזמון מבוקר ע"י שורה 21∅. יתכן ואתה מצפה כי PAUSE 5∅ יגרום לתקתוק פעם בשניה — אך אל נא תשכח כי גם החישוב עצמו אורך זמן שאותו יש לקחת בחשבון. הדבר מתקבל בצורה הטובה ביותר בשיטת ה"נסוי־והשגיאה", ע"י הרצת המחשב מול שעון אמיתי וכוון שורה 21∅ עד להתאמה ביניהם. (אינך יכול לבצע זאת בדיוק רב מאד; הכוונון של מסגרת בשניה הוא 2% — או מחצית השעה במשך יום שלם).

קיימת דרך מדויקת יותר למדידת זמן. היא משתמשת בתוכנם של מיקומים מסוימים בזכרון. המידע האגור בהם נשלף בעזרת הצהרת PEEK. פרק 25 יפרט־מה בדיוק מבקשים. הבטוי שבו משתמשים

הוא

(65536*PEEK 23674+256*PEEK 23673+PEEK 23672)/50

והוא מספק את מספר השניות שעברו מאז הדלקת המחשב. (עד לערך — 3 ימים וְ־21 שעות, שאז המחשב חוזר שוב לערך Ø).
להלן תוכנית משופרת לפעולת שעון:

```
10 REM First we draw the clock face
20 FOR n=1 TO 12
30 PRINT AT 10-10*COS (n/6*PI),16+10*SIN (n/6*PI);n
40 NEXT n
50 DEF FN t()=INT ((65536*PEEK 23674+256*PEEK 23673+
   PEEK 23672)/50): REM number of seconds since start
100 REM Now we start the clock
110 LET t1=FN t()
120 LET a=t1/30*PI : REM a is the angle of the second hand in
    radians
130 LET sx=72*SIN a: LET sy=72*COS a
140 PLOT 131,91: DRAW OVER 1;sx,sy: REM draw hand
200 LET t=FN t()
210 IF t<=t1 THEN GO TO 200: REM wait until time for next
    hand
220 PLOT 131,91: DRAW OVER 1;sx,sy: REM rub out old hand
230 LET t1=t: GO TO 120
```

השעון הפנימי בו משתמשת שיטה זו צריך להיות בעל דיוק העולה על 0.01% כל עוד שהמחשב מריץ את התכנית, או 1Ø שניות במשך יממה; אולם היא עוצרת באופן זמני בכל עת שאתה מבצע BEEP, או פעולה קשורה בטייפ־הקסיטות, או משתמש במדפסת או בציוד הקפי אחר אותו אתה יכול לחבר למחשב. כל אלו יגרמו לאיבוד זמן.

המספרים PEEK 23674, PEEK 23673 וְ־PEEK 23672 נשמרים בתוך המחשב ומשמשים למניה של חלקי־50 של שניה. כל אחד הוא בתחום Ø עד 255, והם גדלים בהדרגתיות מ־Ø עד 255; לאחר 255 הם חוזרים שוב לערך Ø.

PEEK 23672 הוא הגדל בקצב מהיר יותר. כל 1/50 שניה הוא גדל בְּ־1. כאשר הוא מגיע לערך 255, השנוי הבא מחזירו לְ־Ø, ובה בעת משנה את ערכו של PEEK 23673 בְּ־1. כאשר (כל 256/50 שניות) PEEK 23673 השתנה מִ־255 עד Ø, הוא משנה את PEEK 23672 בְּ־1. בכך מובן כיצד הבטוי הקודם פועל כפעולת שעון.

כעת, בחן בזהירות: נניח ושלושת המספרים שברשותנו הם Ø (עבור PEEK 23674), 255 (עבור — PEEK 23673) וְ־255 (עבור PEEK 23672). פרושו של דבר שחלפו 21 דקות בקרוב מאז הדלקת המחשב — והבטוי אמור לתת את התוצאה —

(65536 *Ø + 256 * 255 + 255) / 5Ø = 13 1 Ø.7

אולם, חבויה גם סכנה. בפעם הבאה בה מתבצעת מניית 1/50 של שניה, שלושת המספרים ישתנו לְ־1, Ø וְ־Ø. לעתים קרובות, הדבר יקרה כאשר נמצא באמצע חישוב הבטוי: המחשב יעריך את PEEK 23674 כְּ־Ø, אולם — אז הוא ישנה את השנים הנותרים לְ־Ø בטרם יספיק לבצע PEEK עליהם.

התשובה תהיה אז —

(65536*0+256*0+0)/50=0

שהינה טעות־משוועת.
כלל פשוט כדי להמנע מבעיה זו היא לחשב את הבטוי פעמיים, פעם אחר פעם, ולהתייחס לתשובה הגדולה יותר.
באם תבחן בזהירות את התכנית הקודמת תוכל להווכח כי היא מבצעת זאת באופן־מלא.
לפנינו רעיון — שאותו כדאי להפעיל על הכלל. הגדר פונקציות.

```
10 DEF FN m(x,y)=(x+y+ABS (x-y))/2: REM the larger of x and y
20 DEF FN u()=(65536*PEEK 23674+256*PEEK 23673+PEEK
   23672)/50: REM time, may be wrong
30 DEF FN t()=FN m(FN u(), FN u()): REM time, right
```

אתה יכול לשנות את שלושת מספרי־המונים כך שתקבל את הזמן האמיתי, במקום הזמן שחלף מאז הופעל המחשב. לדוגמא, להגדיר את השעה 1Ø.ØØ בבקר, תווכח כי דרושים 18ØØØØØ = 5Ø * 6Ø * 6Ø * 1Ø חלקי־חמישים (1/50) של שניה, וכן כי מתקיים:

18ØØØØØ = 65536 * 27 + 256 * 119 + 64

כדי להגדיר את שלושת הערכים — 27, 119 וְ־64, יש לבצע:

POKE 23674 ,27 :POKE 23673 ,119 :POKE 23672 ,64

במדינות בהן תדר הרשת הוא 60 הרץ — בתוכניות הללו יש להחליף את המספר 5Ø במספר 6Ø.
הפונקציה INKEY $ (שאינה בעלת ארגומנט) קוראת את לוח־המקשים. באם תלחץ על מקש יחיד (או מקש יחיד עם מקש SHIFT — שהוא) אזי תוצאתה תהיה התו שאותו מקש גורם באופן־הפעולה־הרגיל (L); בכל מקרה אחר — תוצאתה היא המחרוזת־הריקה.
נסה את התכנית הבאה, הפועלת כמכונת־כתיבה:

```
10 IF INKEY$ <>"" THEN GO TO 10
20 IF INKEY$ ="" THEN GO TO 20
30 PRINT INKEY$;
40 GO TO 10
```

כאן, שורה 1Ø מצפה שתרים את אצבעך מלוח המקשים ושורה 2Ø מצפה שתקיש על מקש־חדש.
זכור כי שלא בדומה לְ־INPUT, INKEY $ אינה ממתינה לך. לכן אינך נדרש להדפיס ENTER, אך מצד שני באם לא תדפיס דבר, אבדת את ההזדמנות שניתנה לך.

תרגילים:

1. מה יקרה באם תבטל את שורה 1Ø בתכנית מכונת־הכתיבה?
2. דרך נוספת לשימוש בְּ־INKEY $ קשורה בְּ־PAUSE, כמו בתכנית החלופית הבאה — עבור מכונת־כתיבה:

```
1Ø PAUSE Ø
2Ø PRINT INKEY $;
3Ø GO TO 1Ø
```

כדי שזו תפעל, מדוע חיוני כי PAUSE לא תסתיים באם היא מגלה שאתה לוחץ על מקש כבר מהרגע בו החלה?

3. התאם־את תכנית המחוג־המשני כך שהיא תראה־גם את מחוגי השעות והדקות, כשהיא מציירת אותם כל דקה. באם אתה בעל שאפתנות — דאג לכך שמידי רבע־שעה התכנית תדאג להדפיס הודעה כל שהיא, או אף להציג מופע כלשהוא — אתה יכול ליצר את קולות השעון המפורסם מאנגליה "ביג־בן" בעזרת BEEP. (ראה בפרק הבא)

4. (לסדיסטים בלבד) נסה זאת —

```
10 IF INKEY$ ="" THEN GO TO 10
20 PRINT AT 11,14;"OUCH!"
30 IF INKEY$ <>"" THEN GO TO 30
40 PRINT AT 11,14;"   "
50 GO TO 10
```


פרק 19 : BEEP

תמצית:

BEEP

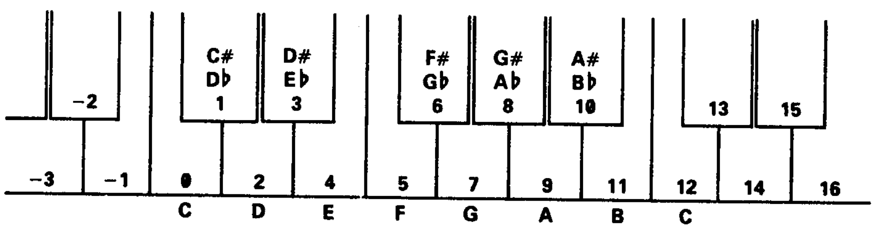
באם לא גילית עד כה כי מחשב ZX-Spectrum קיים רמקול פנימי, קרא שוב את ספרון־המבוא בטרם תמשיך עם פרק זה.

השמעת צליל ברמקול נעשית בעזרת ההצהרה BEEP, שצורתה:

BEEP משך־הזמן, גובה־הצליל

כאשר, כמו בכל ההצהרות האחרות, 'משך־הזמן' ו'גובה־הצליל' מיוצגים על־ידי בטוים־מספריים. משך־הזמן מוגדר בשניות, וגובה־הצליל מוגדר בחצאי־טון מעל C־המרכזי, כשבעזרת מספרים שליליים מייצגים חצאי־טונים מתחתיו.

לפנינו סכימה המציגה את ערכי גובה־הצליל בסולם התוים עבור אוקטבה אחת־בפסנתר:



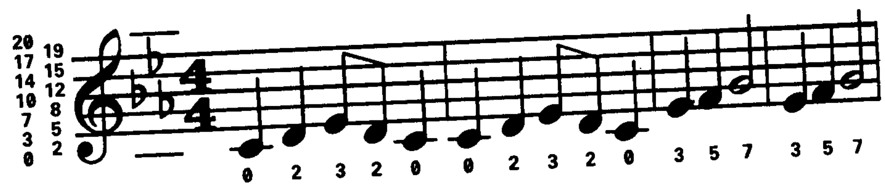
לקבלת תוים גבוהים או נמוכים יותר עליך לחבר או לחסר 12 לכל אוקטבה בה אתה עולה או יורד. באם לפניך נמצא פסנתר בעת שאתה מתכנת נעימה, סכימה זו היא כל שתצטרך כדי להגדיר את ערכי גובה־הצליל. באם, בכל אופן, אתה מעתיק ישירות ממוסיקה כתובה כלשהיא, אנו מציעים שתשרטט סכימה של הקטע עם ערכי גובה־הצליל כתובים מול כל שורה וכל רווח, תוך לקיחת המפתח־בחשבון. לשם דוגמא, הדפס:

```
10 PRINT "Frere Gustav"
20 BEEP 1,0: BEEP 1,2: BEEP .5,3: BEEP .5,2: BEEP 1,0
30 BEEP 1,0: BEEP 1,2: BEEP .5,3: BEEP .5,2: BEEP 1,0
40 BEEP 1,3: BEEP 1,5: BEEP 2,7
50 BEEP 1,3: BEEP 1,5: BEEP 2,7
60 BEEP .75,7: BEEP .25,8: BEEP .5,7: BEEP .5,5: BEEP .5,3:
   BEEP .5,2: BEEP 1,0
70 BEEP .75,7: BEEP .25,8: BEEP .5,7: BEEP .5,5: BEEP .5,3:
   BEEP .5,2: BEEP 1,0
80 BEEP 1,0: BEEP 1,-5: BEEP 2,0
90 BEEP 1,0: BEEP 1,-5: BEEP 2,0
```

כאשר תריץ זאת, אתה אמור לשמוע את "מרש־ההלוויה" מן הסימפוניה הראשונה של מאהלר. הבה נניח כי הנעימה שלך כתובה במפתח C־מינור, כמו הסימפוניה של מאהלר. ההתחלה נראית כך:



אתה יכול לרשום את ערכי גובה־הצליל של התוים באופן הבא:



הוספנו כאן שני קוי־עזר, לשם מידה. שים לב כיצד Eבְּמול בסימני־המפתח משפיע לא רק על הָ־E שבמרווח העליון, בהנמיכו אותו מְ־16 לְ־15, אלא גם על הָ־E בקו התחתון — בהנמיכו אותו מְ־4 לְ־3. כעת, צריך להיות קל מאד למצוא את ערך — 'גובה־הצליל' של כל תו בקטע המוסיקלי.
באם רצונך לשנות את המפתח של הקטע, הדרך הטובה היא להגדיר משתנה — key ולהכניס לתוכנית + Key לפני כל ערך 'גובה־צליל': אזי, השורה השניה תקבל את הצורה:

```
2Ø BEEP 1  ,key+Ø:BEEP 1 ,key+2:BEEP. 5 ,key+3:BEEP .5, key+2:
BEEP 1 ,key + Ø
```

בטרם תריץ תוכנית, אתה חייב להגדיר את ערכו של key:Ø עבור C־מינור, 2 עבור D־מינור, 12 עבור C־מינור באוקטבה אחת גבוהה יותר, וכך הלאה. תוכל להביא את המחשב להתאמה עם כלי אחר ע"י כוון key בעזרת ערכים לא־שלמים.
עליך לקבוע את פרקי־הזמן עבור התוים. מאחר וקטע זה הוא איטי למדי, קבענו פרק זמן של שניה לרבע־תו (crotchet), ואת קבענו לפי כלל זה — מחצית השניה עבור שמינית־תו, וכך הלאה.

הדרגת המשתנה crotchet — האוגר את משך רבע־התו ומגדיר את פרקי־הזמן בהתאם לכך, הינה גמישה יותר. שורה 2Ø תראה אז:

```
20 BEEP crotchet,key+0: BEEP crotchet,key+2: BEEP crotchet/
   2,key+3: BEEP crotchet/2,key+2: BEEP crotchet,key+0
```

(לבטח תרצה לקרוא למשתנים crotchet וְ־key בשמות קצרים יותר).
ע"י קביעת ערכים מתאימים לְ־crotchet, תוכל לשנות את מהירות הקטע.
זכור, כי מאחר וקיים רק רמקול אחד במחשב, נתן לנגן אך ורק תו אחד בו־זמנית, ובכך אתה מוגבל לנעימות ללא־הרמוניות. באם תשאף ליותר — פצח קולך בשיר...
נסה לתכנת בעצמך נעימות. בחר נעימה פשוטה, תוך שאתה מסתייע בכלי פשוט כמו משרוקית — ונתח באמצעותו את הנעימה המבוקשת. אתה יכול ליצור תרשים המראה את ערכי 'גובה־הצליל' לכל תו שנתן להפיקו מכלי זה.
הדפס את השורה הבאה —

```
FOR n = Ø TO 1ØØØ :BEEP .5 ,n :NEXT n
```

אשר תגרום להשמעת תוים במהירות המקסימלית האפשרית ולאחר מכן תעצר עם דווח שגיאה — B integer out of range. תוכל להדפיס את n כדי לדעת עד כמה גבוה הגיע.
נסה זאת, אלא בכוון ההפוך — לתוים הנמוכים. התוים הנמוכים ביותר ישמעו כְּ־קליקים; למעשה התוים הגבוהים גם הם בנויים כקליקים באותו אופן, אלא שהם מהירים יותר מכדי שאוזן האדם תוכל להבחין ביניהם.
רק התחום האמצעי של התוים מיועד למעשה למוסיקה; התוים הנמוכים נשמעים כקליקים והגבוהים דקים מדי ונוטים להיות צורמים.
הדפס את השורה הבאה:

```
1Ø BEEP .5,Ø :BEEP .5,2 :BEEP .5,4 :BEEP .5,5 :BEEP .5,7 :
BEEP .5,9 :BEEP .5,11 :BEEP .5 ,12 :STOP
```

המציגה את סולם C־מז'ור המשתמש בכל התוים ה'לבנים' בפסנתר החל מ־C המרכזי ועד ל־C הגבוה־הקרוב. האופן בו סולם זה מכוון זהה בדיוק כבפסנתר, והוא קרוי מקצב־שלם מאחר ומרווח 'גובה־הצליל' של חצי־טון נשמר שווה לכל אורך הסולם. כנר, בכל אופן, ינגן את הסולם באופן שונה במקצת, על ידי התאמת התוים כך שישמעו נעימים לאוזן. הוא יכול לבצע זאת על־ידי הזזת אצבעותיו לאורך המיתר, דבר שנגן הפסנתר אינו יכול לבצע.
הסולם הטבעי, שאותו מנגן הכנר, נשמע כך:

```
2Ø BEEP .5 ,Ø :BEEP .5 ,2.Ø39: BEEP .5 ,3.86 :BEEP .5 ,4.98:
BEEP .5,7.Ø2 :BEEP .5 ,8.84 :BEEP .5 ,1Ø.88 :BEEP .5 ,12 :STOP
```

אתה עשוי להיות בעל יכולת להבחין בהבדלים בין שני אלו ועשוי גם שלא להיות בעל יכולת כזו; ישנם אנשים בעלי יכולת שכזו; ההבדל הראשון הנכר ביניהם הוא שהתו השלישי נמוך במקצת בסולם הטבעי. באם אתה מוסיקאי מקצועי, יתכן ותרצה לתכנת נעימות בעזרת הסולם הטבעי במקום הסולם בעל המקצבים־השלמים. החסרון בו הוא שלמרות שהוא פועל בצורה יפה בסולם C, עבור מפתחות אחרים הוא פועל בצורה־פחות טובה — לכל אחד מהם קיים הסולם הטבעי שלו — ובכמה מפתחות הוא אף פועל בצורה גרועה מאד. סולם המקצבים־השלמים משתנה רק במעט ופועל בצורה זהה בכל המפתחות.
הדבר אינו מהווה בעיה במחשב, מאחר ואתה יכול להעזר במשתנה key אשר נתן להוסיף לו ערך מספרי כדי להתאים את המפתח.
מוסיקות שונות — במיוחד הודית — משתמשות במרווחי 'גובה־צליל' הקטנים מחצי־טון. תוכל

לתכנתן בהצהרות BEEP ללא כל בעיה; לדוגמא לרבע־הטון מעל C־המרכזי קיים ערך גובה הצליל — .Ø.5

אתה יכול לגרום לכך שלוח־המקשים ישמיע צליל אחר במקום הרגיל — על ידי:

POKE 236Ø9 ,255

המספר השני מגדיר את משך־הצליל (נסה ערכים שונים בתחום Ø עד 255). כאשר הערך הוא Ø, משך הזמן הוא כה קצר עד כי הוא נשמע כקליק־עדין.

באם אתה מעונין להפיק יותר מצלילים מהמחשב, כגון שמיעת הצליל אותו גורמת BEEP בעזרת אמצעי אחר־מלבד הרמקול הפנימי, הרי עליך לדעת כי הצליל מופיע בשני השקעים 'MIC' וְ־'EAR'. בשקע 'EAR' הוא יהיה ברמה גבוהה יותר אך מלבד זאת הם זהים. תוכל להשתמש בו כדי לחבר אוזניות למחשב. חבור שכזה אינו מפסיק את השמעת הצלילים ברמקול הפנימי. באם ברצונך ל'הרעיש עולמות' במוסיקה אותה תיצור תוכל לחבר את המחשב למגבר — בחיבור לשקע 'MIC' — או אף להקליט את יצירותיך על טייפ ולשמרן.

לא תוכל לגרום נזק למחשב גם אם שקעים אלו 'EAR' וְ־'MIC' יתקצרו משום־מה, ולכן אל תחשוש מלנסות — כדי למצוא מהי התפוקה המשביעה את רצונך.

תרגיל:

1. רשום את תכנית הסימפוניה של מאהלר מחדש, בעזרת לולאות FOR-NEXT.
תכנת את המחשב כך שישמיע לא רק את 'מרש־ההלויה' אלא את כל הסימפוניה הראשונה של מאהלר.

פרק 20: שמירת תכניות - על טייפ

תמצית:

MERGE ,VERIFY ,SAVE ,LOAD

השיטות הבסיסיות לשימוש בטייפ־קסיטות כדי לבצע SAVE, LOAD, כן MERGE על תוכניות — מוזכרות בספרון־המבוא. פרק זה מתבסס על האמור בספרון־המבוא ולכן יש לקרוא בו קודם ואף להתנסות באמור שם בטרם נתן להמשיך בקריאת פרק זה.

ראינו כי LOAD משמיטה את התכנית הקודמת ואת המשתנים הקודמים שהיו במחשב לפני הטענת (loading) תכנית חדשה עם משתנים חדשים; קיימת הצהרה נוספת MERGE שאינה גורמת לכך. MERGE משמיטה אך ורק תכנית או שורה קודמת או משתנה קודם באם הדבר נחוץ מאחר ובחדשים מופיע אותו מספר שורה או שם משתנה. הדפס את תכנית ה'קוביה' שבפרק 11 ושמרה על טייפ תחת השם "dice"; כעת הכנס והרץ את התכנית הבאה:

```
1 PRINT 1
2 PRINT 2
10 PRINT 10
20 LET x=20
```

ולאחר מכן המשך כאילו למטרת אימות, אך בהחלפת "VERIFY "dice בהצהרה —

MERGE "dice"

באם תערוך רישום של התכנית (listing) תוכל לראות כי שורות 1 וְ־2 נותרו ללא שנוי, אולם שורות 1Ø וְ־2Ø הוחלפו באלו מתכנית־הקוביה, x גם הוא נותר ללא שנוי (נסה — PRINT x)

ראית עד כה את הצורות הפשוטות של ארבע ההצהרות המשמשות לפעולה עם טייפ־הקסיטות:

SAVE — מקליטה תכניות ומשתנים על טייפ.

VERIFY — משווה את התכנית והמשתנים המוקלטים בטייפ מול אלו הנמצאים כבר במחשב.
LOAD — מנקה את המחשב מכל התכניות והמשתנים הקודמים ומחליפה אותם בחדשים מהקסיטה.
MERGE — דומה לְ־LOAD מלבד העובדה שהיא מוחקת רק תכניות, שורות או משתנים מאחר ומספר־השורה או שם־המשתנה מופיעים גם בתכנית החדשה.

כל אחת מארבע מילות מקש אלו מלווה במחרוזת: SAVE מלווה בשם התכנית תחתיו היא תשמר בטייפ, ועבור שלושת האחרים בשם התכנית אותה המחשב נדרש לחפש ולמצוא. בעת שהוא מחפש — הוא מדפיס את שמות התכניות בהן הוא נתקל בדרכו.

עבור VERIFY, LOAD וכן MERGE אתה יכול לציין את המחרוזת־הריקה כשם אותו המחשב נדרש לחפש: המחשב אז לא יתיחס לשם, וישלוף את התכנית הראשונה בה הוא יתקל.
גרסה אפשרית לְ־SAVE היא מהצורה:

SAVE מחרוזת LINE מספר

תכנית שנאגרה באמצעות הצהרה שכזו, מוקלטת באופן כזה — שכאשר תקרא בחזרה על ידי LOAD (אך לא MERGE) היא תקפוץ מיד לשורה שמספרה נתון, ולמעשה תתחיל להריץ את עצמה — לבד.
עד כה, סוגי המידע היחידים אותם שמרנו בטייפ היו תכניות עם משתניהן. קיימים שני סוגי מידע אחרים הנתָנים גם־הם לאחסון הקרויים — מערכים (Arrays) ו'בתים' (Bytes).
המערכים מטופלים באופן שונה במקצת:
אתה יכול לאגור מערכים בטייפ בעזרת DATA בהצהרת SAVE:

() שם מערך DATA מחרוזת SAVE

המחרוזת תהיה השם תחתיו מידע זה ישמר בטייפ, בדיוק כמו עבור תכניות רגילות או בתים פשוטים.
שם המערך מגדיר את המערך אותו אתה מבקש לאגור, כך שהוא אות בלבד או אות מלווה בסימול $.
זכור את הסוגרים שלאחריו; אתה עשוי למצוא אותם חסרי תועלת מבחינה לוגית, אך אתה חייב לרשום אותן כדי לפשט את עבודת המחשב.
ודא כי ברור לך ההבדל בין תפקידי שם המערך והמחרוזת. אם תרשום, לשם דוגמא —

SAVE "Bloggs" DATA b ()

אזי SAVE יטול את המערך b מהמחשב ויאגור אותו בטייפ תחת השם "Bloggs".

VERIFY "Bloggs" DATA b()

(כאשר המחשב ימצא
אותו הוא יציין 'Number Array :Bloggs' — מערך מספרי בשם Bloggs) ויבדוק אותו מול המערך b

LOAD "Bloggs" DATA b ()

ימצא את המערך בטייפ, ואחר כך — באם ישנו מקום פנוי בזכרון — ישמיט מערך הקיים שם בשם b, באם אכן קיים מערך כזה, ויטעין את החדש מהטייפ תחת השם b.
אינך יכול להשתמש בהצהרה MERGE עם מערכים האגורים בטייפ.
אתה יכול לאגור מערך תוים (מחרוזת) בדיוק באותו אופן. בעת שהמחשב מחפש בטייפ ומוצא אחד שכזה, הוא יודיע 'Character Array' (מערך־תוים) כשלאחריו שם המערך. בעת הטענת מערך־תוים הוא ישמיט מהזכרון לא רק את מערך התוים בעל אותו שם, אלא גם כל מחרוזת באותו שם.
אגירת בתים (Bytes) משמשת לאגירות פיסות מידע ללא כל חשיבות למה משמש מידע זה — הוא יכול להיות תמונת טלויזיה, סימול בגרפיקה־מוגדרת־מפעיל או כל דבר אחר שהמצאת בעצמך, בעזרת CODE:

SAVE "picture" CODE 16384 ,6912

יחידת אגירת המידע בזכרון היא בית (שהוא מספר בתחום ∅ עד 255); לכל בית מוגדרת כתובת (שהיא מספר בתחום ∅ עד 65535).
המספר הראשון לאחר CODE היא כתובת הבית הראשון אותו יש לאגור בטייפ והמספר השני הוא מספר הבתים אותם יש לשמור. בדוגמא שלנו, 16384 היא כתובת הבית הראשון בקובץ־התצוגה (הקובץ המכיל את תמונת הטלויזיה) וְ־6912 הינו מספר הבתים שבו, כך שלמעשה אנו שומרים העתק של מסך הטלויזיה; נסה זאת. השם "picture" פועל באותו אופן כמו שמות תוכניות.
כדי להטעין זאת חזרה, הדפס

LOAD "picture" CODE

אתה רשאי להגדיר את המספרים לאחר CODE, בצורה —

LOAD "poctire" CODE התחלה, אורך

כאן האורך הוא רק מטעמי בטחון; כאשר המחשב מוצא את הבתים בטייפ המתאימים לשם שהוגדר הוא ימנע מלהטעינם באם אורכם בטייפ עולה על האורך לו אתה מצפה (כפי שהגדרת ב'אורך') — מאחר וקיים מידע רב משציפית, ויתכן שבעת ההטענה הוא יכתב על קטעים בזכרון שאתה נזקק להם — ואשר אינך רוצה כי ימחקו. המחשב יציג דווח שגיאה R Tape loading error. אתה יכול לוַתר על ה'אורך' ואז המחשב יקרא את כל הבתים הקיימים.
ה'התחלה' מציגה את הכתובת אליה יוטען הבית הראשון — וכתובת זו יכולה להיות שונה מהכתובת ממנה הועתק הבית הזה לטייפ, כאשר אם שניהם זהים אתה יכול להשמיט — ערך זה מהצהרת LOAD.
6912, 16384, CODE הוא כה יעיל לאגירת והטענת התמונה עד כי תוכל להחליפו בְ־$ SCREEN

לשם דוגמא:

SAVE "picture" SCREEN$
LOAD "picture" SCREEN$

זהו מקרה נדיר שבו VERIFY לא יפעל — VERIFY כותבת את השמות אותם היא מוצאת בטייפ, כך שעד אשר היא מגיע לאימות, קובץ התצוגה השתנה והאימות נכשל. בכל המקרים האחרים עליך להשתמש בְ־VERIFY בכל עת שאתה משתמש בְ־SAVE.
בהמשך, מוגשת תמצית שלמה של ארבע ההצהרות הללו.
'שם' פרושו — בטוי־מחרוזת כלשהוא והוא מתייחס לשם תחתיו נאגר המידע בקסיטה. הוא צריך לכלול תוים מקוד ASCII, שהם רק העשרה הראשונים נמצאים בשימוש.
קיימים ארבעה סוגי מידע הנתנים לאגירה על טייפ: תכניות ומשתנים (יחדיו), מערכים מספריים, מערכי־תוים ובתים עצמאיים.
בעת שְ־VERIFY, LOAD וְ־MERGE סורקים בטייפ למציאת מידע תחת שם נתון וסוג מוגדר — הם מדפיסים במסך את סוגי ושמות כל המידע בו הם נתקלים בזמן החיפוש. סוג המידע מוצג כְּ־'Program' (תכנית), 'Number Array' (מערך מספרי), Character Array (מערך־תוים) או 'Bytes' (בתים). באם השם היה המחרוזת־הריקה, המידע הראשון מהסוג שהוגדר נשלף מן המחשב — בהתעלם משמו.

:SAVE
שומרת מידע על טייפ תחת שם נתון. דווח שגיאה F יתקבל כאשר השם־ריק או כולל 11 או יותר תוים.
SAVE מציגה תמיד את ההערה: Start tape ,then press any key (הפעל את הטייפ, ולאחר זאת הקש על מקש כלשהוא) וממתינה להקשה על מקש בטרם תתחיל לאגור את המידע.
1. תכנית ומשתנים:

SAVE שם

הוא האופן הרגיל.

SAVE שם LINE מספר שורה

אוגרת את התכנית והמשתנים כך — שלאחר LOAD מתבצעת אוטומטית ההוראה —

GO TO מספר שורה הנ"ל

SAVE שם LINE — זהה להצהרה — SAVE שם LINE 1, כך שלאחר שהתכנית תוטען היא תריץ את עצמה מן ההתחלה.

2. בתים:

SAVE שם CODE התחלה, אורך

אוגרת מספר בתים השווה ל'אורך' מכתובת השווה ל'התחלה'.

SAVE שם SCREEN $

הזהה להצהרה —

SAVE שם CODE 16384,6912

אוגרת את מסך הטלויזיה בטייפ.

3. מערכים:

SAVE שם DATA אות ()

או —

SAVE שם DATA אות $ ()

אוגרות את המערך ששמו הוא האות, או האות בצרוף $ — שלהן אין, כזכור, כל תלות ב'שם'.

VERIFY:

בודקת את המידע שבטייפ כנגד המידע הנמצא כבר בזכרון. אי הצלחה באימות מביאה לדווח R Tape loading error

1. תכנית ומשתנים:

VERIFY שם

2. בתים:

VERIFY שם CODE התחלה, אורך

באם מספר הבתים המתאימים ל'שם' — גדול מה'אורך' במספר, מתקבל דווח שגיאה — R. במקרים אחרים היא בוחנת אותם בהשוואה לאלו שבזכרון החל מכתובת ה'התחלה'.

VERIFY שם CODE התחלה

בוחנת את הבתים המתאימים ל'שם' ביחס לאלו בזכרון, המתחילים בכתובת ה'התחלה'.

VERIFY שם CODE

בוחנת את הבתים ב'שם' שהוגדר־כנגד אלו שבזכרון, החל מהכתובת ממנה נאגר הבית הראשון שעל הטייפ.

VERIFY שם SCREEN $

זהה להצהרה —

VERIFY שם CODE 16384,6912

אולם בוודאות גמורה, לא תצליח לבצע את האימות.

3. מערכים:

VERIFY שם DATA אות ()

או —

VERIFY שם DATA אות $ ()

וחנות את שם המערך כנגד ה'אות', או ה'אות' בלויי $, שבזכרון.

LOAD:

1. תכנית ומשתנים:

LOAD שם

משמיטה את התכנית והמשתנים הקודמים ומטעינה את התכנית ומשתניה המתאימה ל'שם' מהטייפ; באם התכנית נאגרה באמצעות SAVE שם LINE, תתבצע קפיצה אוטומטית לשורה המוגדרת והרצת התכנית ממקום זה.
דווח השגיאה 4 Out of memory יתקבל באם אין מקום לתכנית החדשה ומשתניה. במקרה זה — התכנית הקודמת ומשתניה לא יושמטו מהזכרון.

2. בתים:

LOAD שם CODE התחלה, אורך

באם מספר הבתים המתאימים ל'שם' גדול מה'אורך' במספר — יתקבל דווח שגיאה R. בכל מקרה אחר, הם יטענו לזכרון החל מכתובת ה'התחלה' — תוך כתיבה גם על שורות שהיו כתובות מקודם.

LOAD שם CODE התחלה

מטעינה את הבתים המתאימים ל'שם' מהטייפ בזכרון, החל מכתובת ה'התחלה' — תוך כתיבה גם על שורות שהיו כתובות מקודם.

LOAD שם CODE

מטעינה את הבתים המתאימים ל'שם' מהטייפ בזכרון, החל מהכתובת ממנה הבית הראשון שנאגר בטייפ — הוטען, תוך כתיבה גם על שורות שהיו כתובות מקודם.

3. מערכים:

LOAD שם DATA אות ()

או

LOAD שם DATA אות $ ()

משמיטות כל מערך קיים הקרוי 'אות' — או אות בלויי $, ויוצרות חדשים מאלו שנאגרו בטייפ. דווח השגיאה 4 Out of memory יתקבל באם לא נותר מקום עבור המערכים החדשים. מערך קודמים לא יושמטו.

MERGE:

מטעינה מידע חדש מהטייפ ללא השמטת מידע קודם מהזכרון.

1. תכנית ומשתנים:

MERGE שם

מאחדת את התכנית ששמה הוגדר עם זו שקיימת כבר בזכרון, תוך כתיבה על שורות התוכנית או המשתנים של התכנית הישנה אשר מספרי השורות שלה או שמות משתניה מתאימים לאלו

הקיימים בתכנית החדשה.

דווח השגיאה 4 Out of memory מתקבל כאשר אין מקום מספיק בזכרון — לתכנית הישנה ומשתניה ולתכנית החדשה ומשתניה — בשלמותם.

2. בתים:

בלתי־אפשרי

3. מערכים:

בלתי אפשרי.

תרגילים:

1. ערוך קסיטה שבּה התכנית הראשונה תציג את רשימת שאר התכניות שבקסיטה, תבקש ממך לבחור תכנית ולבסוף — תטעין אותה.
2. הבא את גרפיקת כלי־השחמט מפרק 14 ואחר הדפס NEW: הם יעמדו בכך. אולם, הם לא יעמדו בפני כבוי המחשב; באם ברצונך לשמרם, עליך לשמרם בטייפ בעזרת SAVE עם CODE. הדרך הקלה ביותר לאגור את כל 21 הסימולים הגרפיים — היא בעזרת:

SAVE "chess" CODE USR "a", 21*8

— שלאחריו

VERIFY "chess" CODE

זוהי הדרך לאגירת הבתים — לשם אגירת התמונה. כתובת הבית הראשון אותו יש לאגור היא "USR a",
הכתובת של הראשון מבין שמונת הבתים המגדירים את תבנית הסימול־הגרפי (מוגדר־המפעיל) הראשון, ומספר הבתים שיש לאגור הוא 21*8 — שמונה בתים לכל אחד מ־21 הסימולים הגרפיים.
כדי לטעון חזרה — תשתמש בדרך כלל בהצהרה —

LOAD "chess" CODE

אולם, באם תטעין את התכנית חזרה למחשב ZX Spectrum בעל קבולת זכרון אחרת, או באם שינית את הכתובות השונות של הגרפיקה־מוגדרת־המפעיל לכתובות חדשות (אתה חייב לבצע זאת במכוון בעזרת טכניקה מתקדמת יותר), עליך להיות זהיר יותר ולהשתמש בצורה הבאה:

LOAD "chess" CODE USR "a"

USR מאפשרת את קיום העובדה שהגרפיקה תוטען לכתובת שונה.

פרק 21: המדפסת

תמצית:

LPRINT, LLIST, COPY

הערה: הצהרות אלו אינן סטנדרטיות בשפת BASIC, למרות שׁ־LPRINT נמצאת בשימוש בכמה מחשבים אחרים.

באם ברשותך מדפסת — ZX, ימצאו ברשותך בנוסף לה — הוראות הפעלה. פרק זה מכסה את הצהרות הָ־BASIC הנדרשות להפעיל מדפסת זו.
שתי הראשונות — LPRINT וְ־LLIST הן בדיוק כמו — PRINT וְ־LIST, מלבד העובדה שהן פועלות על המדפסת במקום על המסך. צרופה של האות L הוא מסיבה היסטורית. כאשר הומצאה שפת ה־BASIC היא פעלה רק עם מכונת כתיבה חשמלית — ולא עם מסך טלויזיה, ולכן PRINT היה באמת במשמעותו המילולית 'הדפס'. באם כמות המידע שתדפיס — רבה, תיעזר במדפסת־שורה מהירה (Fast line printer) שתחובר למחשב, וההצהרה LPRINT משמעותה: הדפס באמצעות מדפסת השורה (Line).
נסה תכנית זו כדוגמא:

```
10 LPRINT "This program".'
20 LLIST
30 LPRINT '"prints out the character set."'
40 FOR n=32 TO 255
50 LPRINT CHR$ n;
60 NEXT n
```

ההצהרה השלישית, COPY, מדפיסה העתק של מסך הטלויזיה. לדוגמא — הדפס LIST כדי לראות על המסך את רישום תכנית זו ואחר הדפס —

COPY

שים לב כי COPY אינה פועלת עם הרשימות אותן מציג המחשב אוטומטית, מאחר ואלו מתרוקנים בכל עת שמבוצעת פקודה. אתה צריך להשתמש ב־LIST תחילה או להעזר ב־LLIST ולשכוח בכך מְ־COPY.
אתה יכול לעצור את המדפסת בעת פעולתה — בעזרת המקש BREAK (CAPS SHIFT עם SPACE)
באם אתה מבצע הצהרות אלו ללא מדפסת המחוברת למחשב, המחשב יאבד את כל הפלט וימשיך הלאה עם ההצהרה הבאה. נסה זאת:

```
10 FOR n=31 TO 0 STEP -1
20 PRINT AT 31-n,n; CHR$ (CODE "0"+n);
30 NEXT n
```

תתקבל תבנית של תוים הנמשכים באלכסון כלפי מטה — מהפינה הימנית העליונה עד לתחתית המסך, כאשר התכנית תפנה אליך בשאלה — האם אתה מבקש 'לגלגל' (scroll) את המסך.
כעת שנה את n, AT 31-n בשורה 2Ø ל־TAB n. ביצועיה של התכנית יהיו בדיוק כמקודם.
כעת שנה את PRINT שבשורה 2Ø ל־LPRINT. הפעם לא תופיע השאלה ?scroll, אשר אין לה כל משמעות במדפסת, והתבנית תמשיך עם האותיות F עד O.
כעת שנה את TAB n ל־n, AT 31-n, עדיין עם LPRINT. כעת תתקבל רק שורה אחת של סימולים.
הסיבה להבדל שנוצר נעוצה בעובדה שהפלט כתוצאה מ־LPRINT אינו מודפס מיד, אלא תחילה הוא מתארגן בחוצץ (buffer) — עד לאגירת שורה שלמה של תוים אותם המחשב מבקש לשלוח למדפסת בעת ביצוע גישה למדפסת. ההדפסה מתבצעת:

(i) כאשר החוצץ מלא,
(ii) בעקבות הצהרת LPRINT שאינה מסתיימת בפסיק או בפסיק־נקודה,
(iii) כאשר פסיק, גרש או פרט TAB דורש שורה חדשה, או
(iv) בסוף התכנית, באם נותרו תוים שלא הודפסו עדיין.

(iii) — מסביר מדוע התכנית שלנו עם TAB פועלת באופן בו חזינו. לגבי AT, מתעלמים ממספר השורה, ומיקום ה־LPRINT (כמו מיקום ה־PRINT, אלא שהמדובר במדפסת במקום במסך) מקבל את ערך מספר העמודה. פריט AT — אינו יכול לגרום לשורה להשלח למדפסת.

תרגיל:

1. הדפס גרף של הפונקציה SIN ע"י הרצת התכנית שבפרק 17 ושימוש ב־COPY.

פרק 22: ציוד הקפי נוסף:

קיים ציוד נוסף אותו תוכל לחבר ל־ZX-Spectrum.
כונן־הדיסקט־הזעיר (ZX-Microdrive) הינו התקן לאגירת מידע רב והוא גמיש הרבה יותר — במובן בו נתן להשתמש בו — מטייפ הקסיטות. ההצהרות עמן הוא פועל — מלבד SAVE, VERIFY, LOAD ו־MERGE הן — PRINT, LIST, INPUT וכן INKEY$.
הרשת (Network) משמשת לחיבור מספר מחשבי ZX Spectrum — כך שהם יכולים ל"שוחח" ביניהם; שימוש אחד לכך הוא השימוש בכונן־דיסקט אחד עבור כולם.
הממשק (Interface) בסטנדרט־RS 232 — הינו צורת חיבור תקנית המאפשרת לך לקשור את המחשב עם לוח מקשים, מדפסות, מחשבים ומכונות שונות נוספות — אף באם הן לא תוכננו במיוחד עבור ZX Spectrum.
הללו ישתמשו במילות־מקש נוספות, הקיימות כבר בלוח המקשים, אך אינן נתנות לשימוש ללא הקשרים — הנוספים: אילו הן — OPEN #, CLOSE #, MOVE, ERASE, CAT וכן, FORMAT.

פרק 23: קלט ופלט:

תמצית:

OUT
IN

המחשב יכול לקרוא מהזכרון ולכתוב (כך לפחות ב־RAM) — בזכרון בעזרת ההצהרת PEEK וְ־POKE. למחשב עצמו אין זה חשוב באם הזכרון הוא ROM, RAM או אף אם קיים זכרון בכלל; הוא יודע רק כי עבורו קיימות 65536 כתובות בזכרון — כשמכל אַחת מהן הוא יכול לקרוא בית (גם אם לתוכנו אין כל משמעות) ולכתוב בית (גם אם יאבד אי־שם). באופן אנלוגי לחלוטין, קיימים 65536 פוֹרְטֵי קלט/פלט (I/o-ports) — המשמשים את המחשב ליצירת תקשורת עם אלמנטים שונים כגון לוח המקשים, המדפסת; אלמנטים אלו נתנים לשליטה על ידי המחשב בעזרת ההצהרה OUT והפונקציה IN.
IN הינה פונקציה הדומה לפונקציה PEEK:

כתובת IN

לפונקציה — ארגומנט אחד, כתובת הפורט ותוצאתה היא הבית אותו היא קראה מפורט זה.
OUT היא הצהרה הדומה ל־POKE:

ערך, כתובת OUT

מעבירה את הערך לפורט שכתובתו הוגדרה בהצהרה. האופן בו מפורשת כתובת זו — תלוי בכלל המחשב; לעתים קְרובות, לכתובות שונות תהיה אותה משמעות. במחשב זה קל — מאד לחוש את הכתובות מאחר והן כתובות בקוד הבינארי בהן הביטים עצמם נוטים להיות בלתי־תלויים זה בזה. קיימים 16 ביטים, שאותם נכנה (בשימוש ב־A במשמעות adress — כתובת)

A15, A14, A13, A12,......, A2, A1, AØ

AØ הוא הביט הראשון, A1 השני, A2 השלישי וכך הלאה. ביטים AØ, A1, A2, A3 וכן A4 הם החשובים. הם בדרך כלל 1, אך אם אחד מהם הוא Ø, משמעות הדבר עבור המחשב כי עליו לבצע משהוא מוגדר. המחשב אינו יכול להתעסק עם יותר מאלמנט אחד בו־זמנית, ולכן לא יותר מאחד מחמשת הביטים האלו רשאי להיות Ø בו־זמנית. מביטים A5, A6 וְ־A7 מתעלמים, כך שאם אתה אשף באלקטרוניקה תוכל להשתמש בהם בעצמך. הכתובת הטובה ביותר לשימוש היא זו שהינה 1 פחות מכפולה של המספר 32, כך שְ־AØ עד A4 כולם 1. הביטים A8, A9 והלאה משמשים לעתים כדי־לספק מידע נוסף.
הבית שנקרא או נכתב מכיל 8 ביטים, ואילו מסומנים בדרך כלל בצורה DØ, D1 ..., D6, D7 (D מהמילה data — מידע).
לפנינו רשימת כתובות של פורטים הנמצאים בשימוש.
קיים אוסף של כתובות קלט (Input) הקורא את לוח־המקשים ואת השקע — EAR. לוח המקשים מחולק ל־8 חצאי־שורות שבכל אחד מהם 5 מקשים:

CAPS SHIFT to **V**	קורא את מחצית השורה	**IN 65278**
A to **G**	קורא את מחצית השורה	**IN 65022**
Q to **T**	קורא את מחצית השורה	**IN 64510**
1 to **5**	קורא את מחצית השורה	**IN 63486**
0 to **6**	קורא את מחצית השורה	**IN 61438**
P to **7**	קורא את מחצית השורה	**IN 57342**
ENTER to **H**	קורא את מחצית השורה	**IN 49150**
SPACE to **B**	קורא את מחצית השורה	**IN 32766**

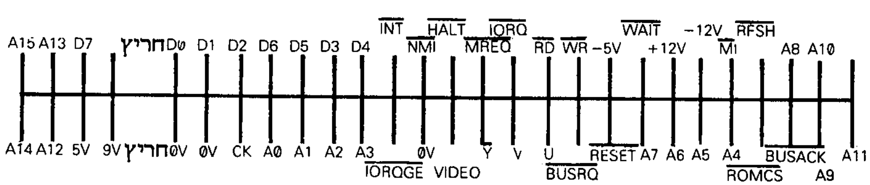
(כתובות אלו הן (255—2↑n) *254+256 עבור n בתום Ø עד 7).
בבית שנקרא, הביטים DØ עד D4 מתאימים לחמשת המקשים במחצית השורה הנתונה — DØ עבור המקש החיצוני, D4 עבור המקש הקרוב למרכז. הביט הוא Ø באם המקש לחוץ, 1 באם לא. D6 הוא הערך בשקע — EAR.
פורט הפלט בכתובת 254 — מפעיל את הרמקול (D4) ואת השקע MIC (D3) וכמו כן מגדיר את צבעי הגבול (DØ, D1, D2).
הפורט בכתובת 251 מפעיל את המדפסת הן בכתיבה והן בקריאה: הקריאה מתבטאת בקביעת העובדה האם המדפסת מוכנה לקבל מידע נוסף, והכתיבה שולחת מידע להדפסה.
הפורטים בכתובות 254, 247 וְ־239 משמשים התקנים הקפיים נוספים שהוזכרו בפרק 22.
הרץ את התכנית הבאה

```
10 FOR n=0 TO 7: REM half-row number
20 LET a=254+256*(255—2↑n)
30 PRINT AT 0,0; IN a: GO TO 30
```

והשתעשע עמה תוך לחיצה על מקשים שונים. באם תשתעמם מחצאי־השורות, הקש BREAK ואחר הדפס —

NEXT n

קוי הבקרה, מידע והכתובת של המחשב חשופים כולם מצידו האחורי של המחשב, כך שלמעשה תוכל לבצע עם הַ־Spectrum כמעט כל דבר שתוכל לבצע עם המיקרו־פרוססור Z80. לעיתים, חומרת המחשב תפריע מלבצע דברים האפשריים עם Z80.
להלן סכימת החבורים שמאחור:



פרק 24 : הזכרון

תמצית :

CLEAR

עמוק, בתוך המחשב, נאגר הכל בבתים (bytes), דהיינו — מספרים בתחום Ø עד 255. אתה עשוי לחשוב כי אגרת את מחיר הצמר או את כתובות ספקי המזון, אולם כל אלו תורגמו לאוסף של בתים — ובתים הם מה שרואה המחשב עצמו.

לכל מקום בו נתן לאגור בית — קיימת כתובתו שהיא מספר בתחום Ø עד FFFF h (כך שהכתובת נתנת לאגירה — כשני בתים), כך שאתה רשאי לראות את הזכרון כשורה ארוכה של תיבות ממוספרות, שכל אחת מהן יכולה להכיל בית אחד. לא כל התיבות זהות. ב־RAM הסטנדרטי בגודל 16K — התיבות מְ־8ØØØh עד FFFFh חסרות כולן. התיבות מְ־4ØØØ h עד 7FFF h הן תיבות RAM, שמשמעותו — שאתה יכול לשנות את תוכנה של כל תיבה, ואילו מְ־Ø ועד 3FFF h הן תיבות ROM — שלהן אין אפשרות לשנות את תוכנן, אלא רק להשתמש בתוכן הקיים, כתיבות בעלות מכסה זכוכית אשר לא נתן לפתחו.

ROM	RAM	לא בשימוש	
Ø	4ØØØh =16384	8ØØØh =32768	FFFFh =65535

כדי לבחון את תוכנה של תיבה — משתמשים בפונקציה PEEK : הארגומנט שלה הוא כתובת התיבה ותוצאתה היא תוכן התיבה. לדוגמא, התכנית הבאה מדפיסה את 21 הבתים הראשונים ב־ROM (ואת כתובותיהם):

```
10 PRINT "Address"; TAB 8; "Byte"
20 FOR a=0 TO 20
30 PRINT a; TAB 8; PEEK a
40 NEXT a
```

בתים אלו יתכן והם חסרי משמעות עבורך, אולם הרכיבים האלקטרוניים במחשב מבינים אותם היטב — כהוראות האומרות מה יש לבצע.

כדי לשנות את תוכנה של תיבה (באם היא RAM), נשתמש בהצהרה POKE שצורתה:

POKE תוכן חדש, כתובת

כאשר, 'כתובות' ו'תוכן חדש' הם שמות לבטויים מספריים. לדוגמא, באם תאמר —

POKE 31ØØØ ,57

הבית שכתובתו 31ØØØ יקבל את הערך החדש 57. הדפס —

PRINT PEEK 31ØØØ

כדי להווכח בכך. (נסה ערכים אחרים כדי להווכח כי אין כל תרמית). הערך החדש חייב להיות בתחום 255 — עד 255+, ובאם הוא שלילי אזי נוסף לו 256.

היכולת לבצע POKE נותנת בידך כוח עצום על פני המחשב באם תדע כיצד להשתמש בו; ויכולת הרסנית עד מאד — באם לא תדע כיצד. קל מאד, על ידי ביצוע POKE בכתובת הלא נכונה וקבלת ערך לא־נכון לאבד תכניות ארוכות מאד שגזלו ממך שעות הדפסה רבות. למזלך, אינך יכול לגרום לנזק כלשהוא למחשב.

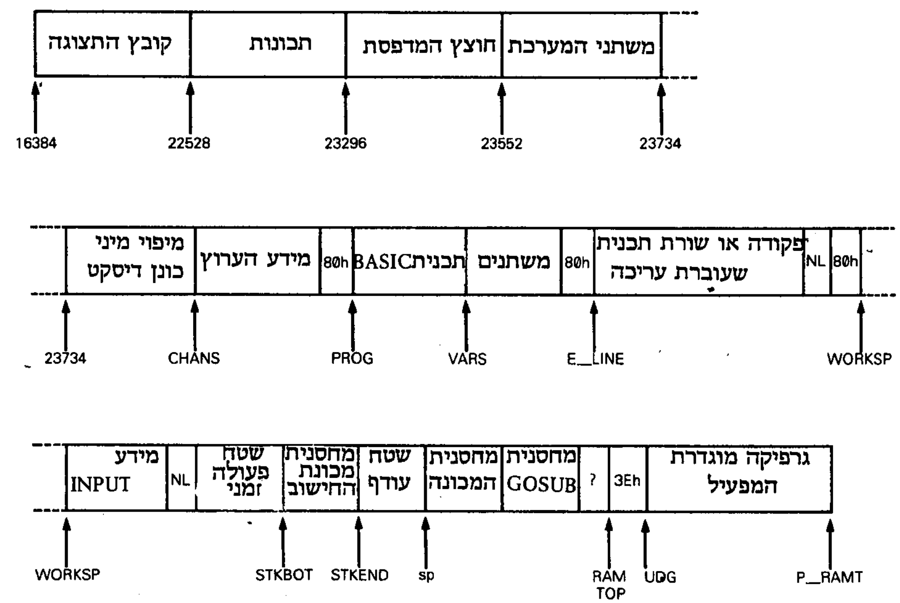
הבה נבחן כעת בפרוט כיצד משתמשים בְ־RAM, אך אל תטרח לקרוא זאת באם אין זה נשמע מעניין. הזכרון מחולק לאיזורים שונים (כפי שמוצג בדיאגרמה הגדולה) למטרת אגירת סוגי מידע שונים. האיזורים הינם גדולים דים עבור המידע אותו הם צריכים לאגור, אך אם תכניס מידע נוסף (למשל ע"י הוספת שורת תוכנית או משתנה) בנקודה מסוימת, נוצר מרווח על ידי הזזת כל שהיה מעל הנקודה — כלפי מעלה. בצורה הפוכה, באם תשמיט מידע מן התכנית, החלק המתאים ינוע כלפי מטה.

קובץ התצוגה (display-file) אוגר את תמונת הטלויזיה, הוא ערוך בצורה מסקרנת, כך שלבטח לא תרצה לבצע עליו את PEEK וְ־POKE. כל מיקום תו במסך מוגדר כמערך ריבועי של 8×8 נקודות כשכל נקודה יכולה להיות Ø ('ניר') או 1 ('דיו'); בעזרת רישום בינארי נתן לאגור את התבנית על ידי 8 בתים, אחד לכל שורת נקודות. מכל מקום, 8 הבתים הללו אינם אגורים בזכרון ביחד. השורות התואמות בְ־32 התוים שבשורה אחת נשמרות ביחד כרצף של 32 בתים, בהתאם לצורת הסריקה של הקרן האלקטרונית על פני מסך הטלויזיה — מצד שמאל לימין. מאחר והתמונה כולה מורכבת מְ־24 שורות בנות 8 סריקות, אתה עשוי לצפות לכך שהסך הכולל־172 סריקות, ישמר בסדר המתאים אחד אחר השני; ובכן טעית. תחילה מתבצעות הסריקות בקצה־העליון של שורות Ø עד 7, לאחר מכן הסריקות הבאות של שורה Ø עד 7 וכך הלאה עד לסריקות בקצה התחתון של שורות Ø עד 7; לאחר מכן חוזר התהליך בשורות 8 עד 15 ולבסוף בשורות 16 עד 23. המסקנה מכך היא שאם אתה רגיל למחשב המשתמש בְ־PEEK וְ־POKE עבור המסך, תצטרך להתחיל להתרגל להשתמש בְ־SCREEN$ וְ־PRINT AT במקומן, או PLOT וְ־POINT.

התכונות הן הצבעים ועוד — לכל מיקום תו, בעזרת הפורמט של ATTR. אלו נאגרות שורה אחר שורה בסדר בו אתה מצפה שישמרו.

חוצץ־המדפסת (printer buffer) אוגר את התוים המיודעים למדפסת.

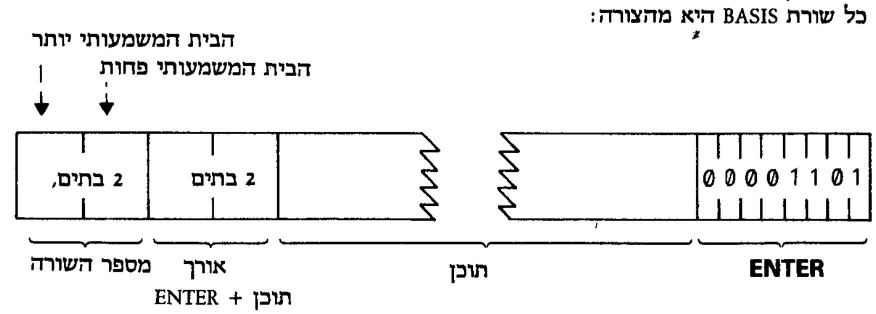
משתני המערכת מכילים חלקי מידע שונים המציינים למחשב באיזה מצב הוא נמצא כעת. הם מופיעים בפרק הבא במפורט, אך לעת עתה שים לב כי ישנם כמה (הקרוים: CHANS, PROG, VARS, E-LINE, ועוד) המכילים את הכתובות של הגדולות בין האיזורים השונים בזכרון. אלו אינם משתני BASIC, ושמותיהם לא יוכרו על ידי המחשב.



מפות כונן הדיסקט הזעיר משמשות רק את כונן־הדיסקט. באופן פעולה נורמלי — אין בהם כל מידע שהוא.

מידע הערוץ מכיל מידע אודות התקני הקלט והפלט, כלומר — לוח המקשים (עם המחצית התחתונה של המסך) המחצית העליונה של המסך, והמדפסת.

כל שורת BASIS היא מהצורה:

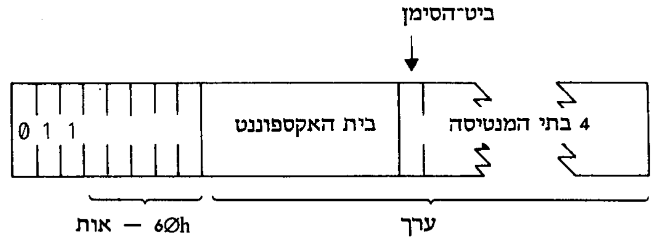


שים לב, כי בניגוד לכל שאר המקרים של מספרים בני שְני־בתים בָ־Z80, מספר השורה כאן נאגר — כאשר הבית המשמעותי יותר נאגר ראשון: זהו הסדר בו אתה מדפיס אותם.

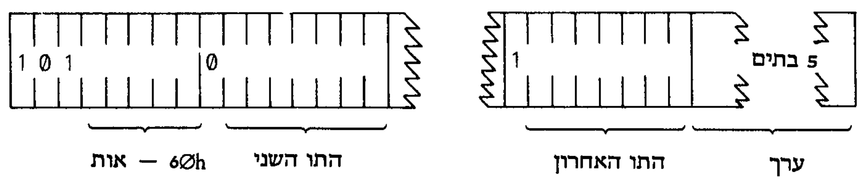
קבוע־מספרי בתכנית מלווה בצורתו הבינארית, בעזרת התו 14 CHR$ שלאחריו 5 בתים עבור המספר עצמו.

למשתנים ישנו פורמט שונה בהתאם למהותם השונה. האותיות בשם צריכות להחשב כמתחילות כאותיות כתיב.

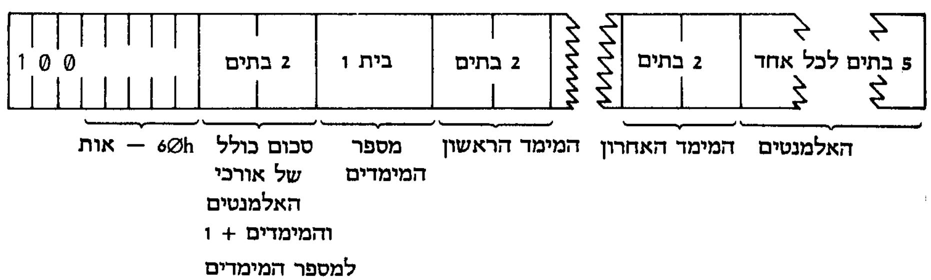
מספר ששמו אות יחידה:



מספר ששמו ארוך מאות אחת



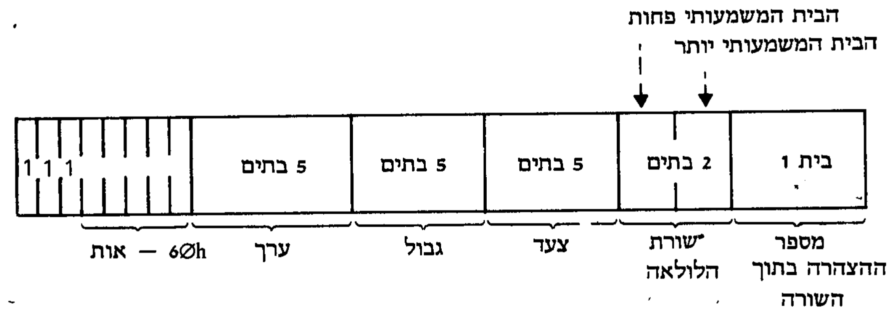
מערך של מספרים:



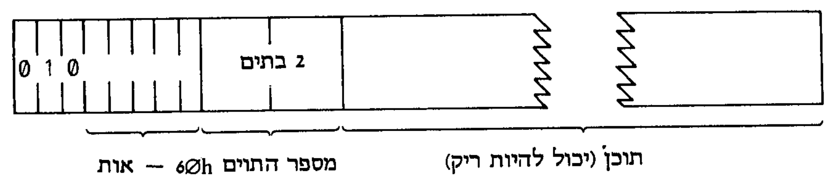
סדר האלמנטים הינו:

תחילה, האלמנטים עבורם התוית הראשונה היא 1
לאחר מכן, האלמנטים עבורם התוית הראשונה היא 2
לאחר מכן, האלמנטים עבורם התוית הראשונה היא 3
וכך הלאה לכל הערכים האפשריים של התוית הראשונה.
האלמנטים שלהם התוית־ראשונה הנתונה זהה, יסודרו לפי התוית השניה, וכך הלאה עד לאחרונה.
כדוגמא, האלמנטים של מערך 3*6 הקרוי b — מפרק 12, נאגרים בסדר הבא:

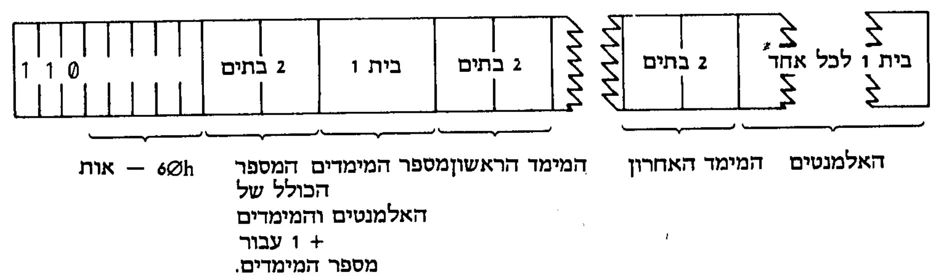
b(1,1) b(1,2) b(1,3) b(1,4) b(1,5) b(1,6) b(2,1) b(2,2) .... b(2,6) b(3,1) b(3,2) ... b(3,6)



מחרוזות:



מערך תוים:



מכונת החישוב (calculator) היא החלק במערכת ה־BASIC המבצע את החישובים אריתמטים, כאשר המספרים עמם הוא מבצע את הפעולות האלו נשמרים בעקר במחסנית מכונת־החישוב.
החלק העודף כולל מרווח — שלא בשימוש.
מחסנית המכונה היא המחסנית בה משתמש ה־Z-80 כדי לשמור כתובות חזרה וכו'.
מחסנית ה־GOSUB הוזכרה בפרק 5.
הבית אליו מצביע RAMTOP הוא בעל הכתובת הגבוהה ביותר בו משתמשת מערכת ה־BASIC. גם NEW — המנקה את ה־RAM מתוכנו, מגיעה רק עד כתובת זו — כך שאינה פוגעת בגרפיקה־מוגדרת־המפעיל. אתה יכול לשנות את כתובת RAMTOP ע"י הכנסת מספר בהצהרת CLEAR:

הצהרה זו —
(i) "מנקה" את כל המשתנים.
(ii) "מנקה" את קובץ התצוגה (כמו — CLS)
(iii) מאפסת את מיקום ה־PLOT לפינה השמאלית — התחתונה.
(iv) מבצעת RESTORE
(v) "מנקה" את מחסנית ה־GOSUB ומציבה אותה בכתובת המתאימה לערך החדש של RAMTOP — בהנחה כי הוא נמצא בין מחסנית מכונת־החישוב והקצה הפיסי של ה־RAM; אחרת RAMTOP ישאר כפי שהיה.

RUN גם היא מבצעת CLEAR, אף כי אינו משנה את RAMTOP
בשימוש ב־CLEAR בדרך זו תוכל להזיז את RAMTOP כלפי מעלה, כדי להכין מקום נוסף ל־BASIC, על ידי כתיבה חוזרת על הגרפיקה־מוגדרת־המפעיל, או שתוכל להניעו למטה כדי להכין מקום ב־RAM המוגן מהשפעת NEW.
הדפס NEW ולאחריו CLEAR 238ØØ כדי לקבל מושג מה קורה למכונה לאחר שהיא מתמלאת.
אחד הדברים הראשונים בו תבחין באם תחל להדפיס תוכנית היא שלאחר זמן המחשב יחדול מלקבל קלט כלשהוא ויתחיל לזמזם. משמעותו של הדבר — שהמחשב מלא, ועליך לרוקנו במקצת. כמו כן קיימות שתי הודעות שגיאה שלהם משמעות קרובה מאד: 4 Memory full וכן G No room for line.
הזמזום מתרחש גם כאשר תדפיס שורה ארוכה — עם יותר מ־23 תוים — המחשב אינו מתעלם ממה שתדפיס אלא שלא תוכל לראות זאת; הזמזום מטרתו להורות לך לחדול.
תוכל לכוון את משך הזימזום בעזרת POKE לכתובת 236Ø8. המשך הרגיל נקבע בעזרת המספר 64.
כל מספר (מלבד Ø) נתן לרישום בצורה —

$$\pm m \times 2^e$$

כאשר —

± הוא הסימן.
m היא המנטיסה — וערכה בין ½ ל־1 (אך־לא 1)
e הוא האקספוננט — שהוא מספר שלם (אפשרי־שלילי)

הבה נניח כי רשמת את m לפי הבסיס־הבינארי. מאחר והוא מספר קטן מ־1 הוא יכלול נקודה בינארית (בדומה לנקודה העשרונית בבסיס 10) ולאחריה שבר בינארי (בדומה לשבר העשרוני); כך שבבסיס בינארי —
1/2 הוא .1
1/4 הוא .Ø1
3/4 הוא .11
1/10 היא ØØØ11ØØ11ØØ11ØØ
המספר שלנו — m, מאחר והוא קטן מ־1, אין ביטים משמאל לנקודה ומאחר והוא גדול מהערך 1/2 — הביט שלאחר הנקודה — יהיה 1.
כדי לאגור את המספר במחשב, נשתמש ב־5 בתים, כדלהלן:
(i) נרשום את 8 הביטים הראשונים של המנטיסה בבית השני (אנו יודעים כי הביט הראשון הוא 1), קבוצת 8 הביטים השניה בבית השלישי, הקבוצה השלישית של 8 הביטים בבית הרביעי והקבוצה הרביעית בבית החמישי.
(ii) החלף את הביט הראשון שבבית השני — שידוע כי הוא 1 — בסימן: Ø עבור חיובי ו־1 עבור שלילי.
(iii) רשום את האקספוננט +128 בבית הראשון. לדוגמא באם המספר שלנו הוא 1/1Ø:

$$1/1\varnothing = 4/5 \times 2^{-3}$$

ולכן המנטיסה — m היא 1100110011001100110011001100 – בבסיס בינארי (מאחר והביט ה־33 במספר הוא 1, אנו נעגל את הביט ה־32 מ־Ø ל־1), והאקספוננט e הוא 3—.
בעזרת שלושת הכללים נקבל את חמשת הבתים:

ה־Ø כאן מצביע על סימן חיובי

0111 1101	0100 1100	1100 1100	1100 1100	1100 1101

המנטיסה־4/5, מלבד העובדה שהביט הראשון צריך להיות 1 עבור אקספוננט. 128+3−

קיימת דרך אלטרנטיבית לאגירת מספרים שלמים בין 65535 — ל־65535+:
(i) הבית הראשון הוא Ø
(ii) הבית השני הוא Ø עבור מספר חיובי ו־FFh עבור שלילי.
(iii) הבתים השלישי והרביעי הם הבתים הפחות והיותר משמעותיים של המספר — בהתאמה (או: המספר + 131Ø72 באם המספר שלילי)
(iv) הבית החמישי הוא Ø.

פרק 25: משתני המערכת

הבתים (bytes) בזכרון בתחום הכתובות 23552 עד 23733 מוגדרים לשימוש מיוחד על ידי המערכת. תוכל לבצע PEEK לגביהם כדי לגלות מה מתרחש במערכת, וכמה מהם נתנים גם לביצוע POKE ביעילות. הם מופיעים ברשימה שבהמשך כשלכל אחד מוסבר השימוש בו.
כל אלו קרויים משתני־המערכת, ולהם שמות, אולם אל תחליפם בטעות במשתנים בהם משתמשת BASIC. המחשב אינו מכיר בשמות כמתייחסים למשתני המערכת, והם נתנים כאן אך ורק כקיצור להקל על הכרתם למשתמשים במערכת.
לקצורים בעמודה 1 המשמעות הבאה:
X אסור שיבוצע POKE על המשתנים מאחר והדבר עלול לפגוע במערכת.
N ביצוע POKE על המשתנה לא יותיר סימן.
המספר שבעמודה 1 הוא מספר הבתים שבמשתנה. עבור שני בתים, הראשון שבהם הוא הפחות משמעותי — להפך ממה שכנראה ציפית. לכן כדי לבצע POKE על ערך V למשתנה בעל שני בתים בכתובת n, השתמש בבטוי —

```
POKE n ,V—256 * INT (V/256)
POKE n+1 ,INT (V/256)
```

וכדי לגלות את ערכו בעזרת PEEK השתמש בבטוי:

```
PEEK n + 256 * PEEK (n+1)
```

הערות	כתובת	שם	תוכן
N8	KSTATE	23552	משמש־בעת קריאת לוח־המקשים.
N1	LAST K	23560	אוגר מקש שנלחץ כעת.
1	REPDEL	23561	הזמן (בחלקי 50 של שניה — או חלקי 60 בצפון אמריקה) שמקש חייב להיות לחוץ בטרם יחזור על עצמו. הוא מתחיל עם — 35, אך אתה יכול לשנותו בעזרת POKE.
1	REPPER	23562	ההשהייה (בחלקי 50 של שניה — או חלקי 60 בצפון אמריקה) בין שתי חזרות סמוכות של החזקת המקש כשהוא לחוץ. ערכו ההתחלתי 5.
N2	DEFADD	23563	כתובת הארגומנטים של פונקציה־מוגדרת־מפעיל באם פונקציה שכזו מחושבת; אחרת — Ø
N1	K DATA	23565	אוגר את הבית השני של בקרת הצבע המוכנס מלוח־המקשים.
N2	TVDATA	23566	אוגר בתי־צבע ובקרות AT וְ־TAB הפונות לטלויזיה.
X38	STRMS	23568	כתובות הערוצים הקשורים לרצף זכרון.
2	CHARS	23606	256 פחות מכתובת אוסף התוים (המתחילה עם מרווח וממשיכה עד לסימן ©). באופן רגיל בְ־ROM, אך אתה יכול לחבר RAM משלך ולדאוג כי CHARS תפנה אליו.
1	RASP	23608	משך צליל האזהרה
1	PIP	23609	משך הקליק עקב הקשה על מקש
1	ERR NR	23610	1 פחות מקוד הדווח. מתחיל בְ־255 (עבור 1 —) כך שְ־PEEK 2361Ø תתן 255
X1	FLAGS	23611	דגלים שונים לבקרת מערכת הַ־BASIC.
X1	TV FLAG	23612	דגלים הקשורים בטלויזיה.
X2	ERR SP	23613	כתובת הפרט במחסנית־המכונה שבו יעשה שימוש בחזרה עקב שגיאה.
X2	LIST SP	23615	כתובת החזרה מרישום (listing) אוטומטי
N1	MODE	23617	מגדיר סמן K, L, C, E או G.
2	NEWPPC	23618	השורה אליה יש לקפוץ.

1	NSPPC	23620	מספר ההצהרה בשורה אליה יש לקפוץ. ביצוע POKE תחילה על NEWPPC ולאחר מכן NSPPC מאלץ קפיצה להצהרה המוגדרת בשורה.
2	PPC	23621	מספר השורה של ההצהרה המבוצעת כעת.
1	SUBPPC	23623	המספר בתוך השורה של ההצהרה המבוצעת כעת.
1	BORDCR	23624	צבע גבול * 8; כמו כן מכיל את התכונות בהן נעשה שימוש באופן נורמלי במחצית התחתונה של המסך.
2	E PPC	23625	מספר השורה הנוכחית (עם סמן תכנית).
X2	VARS	23627	כתובת משתנים.
N2	DEST	23629	כתובת משתנה — בהשמה (assignment).
X2	CHANS	23631	כתובת ערוץ מידע.
X2	CURCHL	23633	כתובת המידע המשמש זמנית עבור קלט ופלט.
X2	PROG	23635	כתובת תכנית בְ־BASIC.
x2	NXTLIN	23637	כתובת השורה הבאה בתכנית.
X2	DATADD	23639	כתובת הסיומת של פרט הַ־DATA האחרון.
X2	E LINE	23641	כתובת הפקודה שהודפסה למחשב.
2	K CUR	23643	כתובת הסמן.
X2	CH ADD	23645	כתובת התו הבא שיפורש: התו לאחר הארגומנט של PEEK, או הַ־NEWLINE בסוף הצהרת POKE.
2	X PTR	23647	כתובת התו שלאחרי הסימן ?.
X2	WORKSP	23649	כתובת שטח הפעולה הזמני.
X2	STKBOT	23651	כתובת תחתית מחסנית־מכונת־החישוב.
X2	STKEND	23653	כתובת תחילת המרווח העודף.
N1	BREG	23655	רגיסטר b של מכונת החישוב.
N2	MEM	23656	כתובת האיזור המשמש כזכרון מכונת־החישוב. (בדרך כלל MEMBOT, אך לא תמיד)
1	FLAGS2	23658	דגלים נוספים.
X1	DF SZ	23659	מספר השורות (כולל שורה ריקה אחת) בחלק התחתון של המסך.
2	S TOP	23660	מספר השורה העליונה בתכנית — ברישום אוטומטי (automatic listing).
2	OLDPPC	23662	מספר השורה אליה קופצת CONTINUE.
1	OSPCC	23664	מספר שורת ההצהרה אליה קופצת CONTINUE
N1	FLAGX	23665	דגלים שונים.
N2	STRLEN	23666	אורך היעד (destination) למחרוזות בהשמה (assignmeut).
N2	T ADDR	23668	כתובת הפרט הבא בטבלאת התחביר (מאד לא שמושית).

2	SEED	23670	הגרעין לפונקצית RND. זהו המשתנה הנקבע על־ידי — RANDOMIZE.
3	FRAMES	23672	מונה מסגרת בן 3 בתים (הפחות משמעותי — תחילה). ערכו מוגדל כל 20ms. ראה בפרק 18.
2	UDG	23675	כתובת הסימול הראשון בגרפיקה — מוגדרת־המפעיל. תוכל לשנות זאת, למשל כדי לשמור מרווח על ידי כך שיהיו פחות סימולים גרפיים שכאלו.
1	COORDS	23677	קואורדינטת x של הנקודה האחרונה ששורטטה.
1		23678	קואורדינטת y של הנקודה האחרונה ששורטטה.
1	P POSN	23679	מספר בתחום 33 העמודות המתאר את מיקום המדפסת.
1	PR CC	23680	הבית הפחות משמעותי בכתובת הבאה עבור LPRINT כדי להדפיס בה (בחוצץ המדפסת).
1		23681	לא בשימוש.
2	ECHO E	23682	מספר בתחום 33־העמודות ומספר בתחום 24־השורות (במחצית התחתונה) של סוף חוצץ הקלט.
2	DF CC	23684	כתובת מיקום ה־PRINT בקובץ־התצוגה.
2	DFCCL	23686	כמו DFCC לחלק התחתון של המסך.
X1	S POSN	23688	מספר בתחום 33 העמודות עבור מיקום ה־PRINT.
X1		23689	מספר בתחום 24 השורות עבור מיקום ה־PRINT.
X2	SPOSNL	23690	כמו S POSN עבור החלק התחתון.
1	SCR CT	23692	מונה את מספר "הגלגולים" (Scrolls): ערכו תמיד 1 יותר ממספר הגלגולים שיבוצעו בטרם יעצור המחשב וישאל — Scroll? באם תבצע POKE על מספר זה עם מספר הגדול מ־1 (למשל 255), המסך יתגלגל עוד ועוד, ללא הצגת השאלה.
1	ATTR P	23693	צבעים נוכחיים־קבועים וכו' (כפי שהוגדרו על ידי הצהרות־הצבע).
1	MASK P	23694	משמש לצבעים־שקופים וכו'. כל ביט שערכו 1 מראה כי ביט התכונה המתאים אינו נלקח מתוך ATTR P אלא ממה שנמצא כבר במסך.
N1	ATTR T	23695	צבעים נוכחיים־זמניים וכו' (כפי שהוגדרו על ידי פרטי הצבע).
			כמו MASK P אלא שהוא זמני.

N1	MASK T	23696	
1	P FLAG	23697	דגלים נוספים.
N30	MEMBOT	23698	שטח הזכרון של מכונת־החישוב; משמש לאגירת מספרים שאינם נתן להכנסה בנוחות למחסנית־מכונת־החישוב.
2		23728	לא בשימוש.
2	RAMTOP	23730	כתובת הבית האחרון בשטח מערכת ה־BASIC.
2	P-RAMT	23732	כתובת הבית האחרון ב־RAM הפיסיקלי.

התכנית הבאה מציגה את 22 הבתים הראשונים של איזור המשתנים:

```
1Ø FOR n=Ø TO 21
2Ø PRINT PEEK (PEEK 23627+256*PEEK 23628+n)
3Ø NEXT n
```

נסה להתאים את משתנה הבקרה n לתיאורים שלמעלה.

כעת שנה את שורה 2Ø לשורה הבאה —

2Ø PRINT PEEK (23755 + n)

המציגה את 22 הבתים הראשונים של איזור התכנית. בחן אותם עם התכנית עצמה.

פרק 26: השימוש בשפת מכונה

תמצית:

USR עם ארגומנט מספרי.

פרק זה נכתב עבור אלו המבינים את 'שפת המכונה' של Z-80, שהיא אוסף ההוראות אותן המיקרופרוססור Z-80 מבין ומבצע. באם אינך מבין נושא זה אך מבקש להכירו, קיימים ספרים רבים על נושא זה. חפש ספרים שכותרתם תהיה מהצורה — 'שפת המכונה של ה־Z80 (או שפת האסמבלי) למתחיל'; באם מוזכר בספר זה גם ZX Spectrum — מה טוב.

תכניות אלו נכתבות בדרך כלל בשפת "אסמבלי", שאינה קשה להבנה עם מעט תירגול. (תוכל לראות את אוסף פקודות האסמבלי בנספח — A). בכל אופן, כדי להריצן במחשב, עליך לקודד את התכנית לרצף של בתים ובצורה זו היא קרויה שפת־מכונה. תרגום זה מבוצע בדרך כלל על ידי המחשב בעצמו, בעזרת תוכנית הקרויה — 'אסמבלר'. במחשב ZX Spectrum לא נבנה אסמבלר — כחלק מהמערכת, אך תוכל בפשטות לרכוש אחד על גבי קסיטה.

בהיעדר כלי זה, תיאלץ לעשות את התרגום במו־ידיך, בתנאי שהתכנית אינה ארוכה מדי.

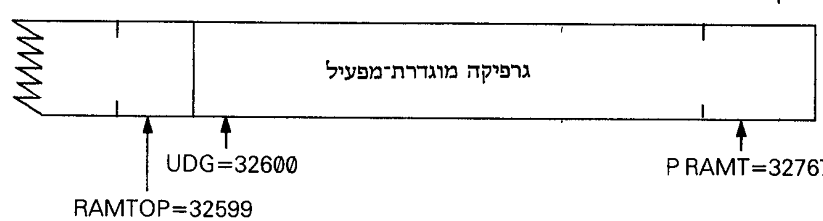
הבה נקח כדוגמא את התכנית:

```
ld bc ,99
ret
```

המטעינה את צמד הרגיסטרים bc בערך 99. הדבר מתורגם לארבעת בתי שפת־המכונה 1, 99, Ø (עבור ld bc ,99) ו־201 (עבור ret). (באם תבדוק בנספח A את הערכים 1 ו־201 תמצא — ld bc ,NN כאשר NN מייצג מספר כלשהוא בן שני־בתים, וכן — ret).

לאחר שקבלת את התכנית בשפת־המכונה, הצעד הבא הוא להכנס לנבכי המחשב. (האסמבלר היה מבצע זאת, כפי־הנראה, אוטומטית). עליך להחליט היכן למקם את התכנית בזכרון, והדבר הטוב ביותר יהיה להותיר מרווח נוסף עבורה בין איזור ה־BASIC והגרפיקה־מוגדרת־המפעיל.

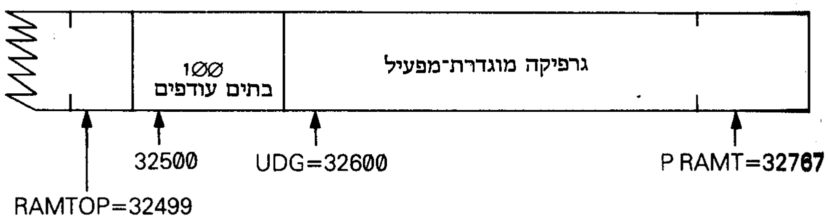
נניח כי ברשותך ZX Spectrum שגודל זכרונו — 16K. כדי להתחיל, בקצהו העליון של ה־RAM — מתקיים:



באם תדפיס —

CLEAR 32499

יתקבל מרווח של 1ØØ בתים (הוגדר ביד־רחבה), שתחילתם בכתובת 325ØØ:



כדי להכניס את התכנית בשפת־המכונה, הרץ תכנית BASIC — הדומה לבאה:

```
10 LET a=32500
20 READ n: POKE a,n
30 LET a=a+1: GO TO 20
40 DATA 1,99,0,201
```

(**תכנית** זו תעצור עם דווח E out of DATA לאחר שמילאה את ארבעת הבתים שהגדרת).

כדי להריץ את שפת־המכונה, העזר בפונקציה USR — אך כעת עם ארגומנט מספרי, בכתובת ההתחלה. תוצאתו היא ערכו של צמד הרגיסטרים bc בעת החזרה מתכנית שפת־המכונה, ולכן באם **תבצע**

PRINT USR 32500

תקבל את התשובה — 99.

כתובת החזרה ל־BASIC מאוחסנת במחסנית בדרך הרגילה, כך שהחזרה היא על ידי הוראת ret של Z80. אינך אמור להשתמש ברגיסטים iy ו־i ברוטינות בשפות מכונה.

תוכל לשמור את התוכנית בשפת־המכונה בקלות בעזרת:

SAVE "שם כלשהוא" CODE 325ØØ ,4

אין כל דרך ישירה — לשמור אותה כך שכאשר תוטען — התכנית תריץ את עצמה — לבדה, אולם **תוכל** לעקוף זאת בעזרת תכנית — BASIC:

```
1Ø LOAD "" CODE 325ØØ ,4
2Ø PRINT USR 325ØØ
```

בצע תחילה —

SAVE "שם כלשהוא" LINE

ואז —

SAVE "XXXX" CODE 325ØØ ,4
LOAD "שם כלשהוא"

יטעינו ואוטומטית יריצו את התכנית ב־BASIC, ואז התכנית ב־BASIC תטעין ותריץ את התכנית בשפת־המכונה.

נספח A: אוסף התוים :

להלן האוסף השלם של התוים הקימים במחשב ZX Spectrum בצרוף קודיהם העשרונים וההקסאדצימלים (בסיס 16). באם תדמיין לעצמך את הקודים כקודי־ההוראות בשפת המכונה של — Z80, אזי בעמודה המתאימה — תמצא את קצורי־הוראות האסמבלי (mnemonics). כפי שלבטח ברור לך, באם אתה מכּיר היטב תחום זה, מספר הוראות של Z80 הנך מורכבות, והן מתחילות עם קידומת CBh או EDh ; העמודות השמאליות מספקות את המידע המתאים.

קוד Code	תו Character	יצוג הקסדצימלי Hex	אסמבלר Z80 — Z80 Assembler	לאחר קידומת CB – after CB	לאחר קידומת ED – after ED
0	לא בשימוש	00	nop	rlc b	
1	לא בשימוש	01	ld bc,NN	rlc c	
2	לא בשימוש	02	ld (bc),a	rlc d	
3	לא בשימוש	03	inc bc	rlc e	
4	לא בשימוש	04	inc b	rlc h	
5	לא בשימוש	05	dec b	rlc l	
6	**PRINT** פסיק	06	ld b,N	rlc (hl)	
7	**EDIT**	07	rlca	rlc a	
8	סמן לשמאל	08	ex af,af'	rrc b	
9	סמן לימין	09	add hl,bc	rrc c	
10	סמן למטה	0A	ld a,(bc)	rrc d	
11	סמן למעלה	0B	dec bc	rrc e	
12	**DELETE**	0C	inc c	rrc h	
13	**ENTER**	0D	dec c	rrc l	
14	מספר	0E	ld c,N	rrc (hl)	
15	לא בשימוש	0F	rrca	rrc a	
16	**INK** control	10	djnz DIS	rl b	
17	**PAPER** control	11	ld de,NN	rl c	
18	**FLASH** control	12	ld (de),a	rl d	
19	**BRIGHT** control	13	inc de	rl e	
20	**INVERSE** control	14	inc d	rl h	
21	**OVER** control	15	dec d	rl l	
22	**AT** control	16	ld d,N	rl (hl)	
23	**TAB** control	17	rla	rl a	

24	לא בשימוש	18	jr DIS	rr b	
25	לא בשימוש	19	add hl,de	rr c	
26	לא בשימוש	1A	ld a,(de)	rr d	
27	לא בשימוש	1B	dec de	rr e	
28	לא בשימוש	1C	inc e	rr h	
29	לא בשימוש	1D	dec e	rr l	
30	לא בשימוש	1E	ld e,N	rr (hl)	
31	לא בשימוש	1F	rra	rr a	
32	space	20	jr nz,DIS	sla b	
33	!	21	ld hl,NN	sla c	
34	"	22	ld (NN),hl	sla d	
35	#	23	inc hl	sla e	
36	$	24	inc h	sla h	
37	%	25	dec h	sla l	
38	&	26	ld h,N	sla (hl)	
39	'	27	daa	sla a	
40	(	28	jr z,DIS	sra b	
41	)	29	add hl,hl	sra c	
42	*	2A	ld hl,(NN)	sra d	
43	+	2B	dec hl	sra e	
44	,	2C	inc l	sra h	
45	−	2D	dec l	sra l	
46	.	2E	ld l,N	sra (hl)	
47	/	2F	cpl	sra a	
48	0	30	jr nc,DIS		
49	1	31	ld sp,NN		
50	2	32	ld (NN),a		
51	3	33	inc sp		
52	4	34	inc (hl)		
53	5	35	dec (hl)		
54	6	36	ld (hl),N		
55	7	37	scf		
56	8	38	jr c,DIS	srl b	
57	9	39	add hl,sp	srl c	
58	:	3A	ld a,(NN)	srl d	
59	;	3B	dec sp	srl e	
60	<	3C	inc a	srl h	
61	=	3D	dec a	srl l	
62	>	3E	ld a,N	srl (hl)	
63	?	3F	ccf	srl a	
64	@	40	ld b,b	bit 0,b	in b,(c)
65	A	41	ld b,c	bit 0,c	out (c),b
66	B	42	ld b,d	bit 0,d	sbc hl,bc
67	C	43	ld b,e	bit 0,e	ld (NN),bc
68	D	44	ld b,h	bit 0,h	neg
69	E	45	ld b,l	bit 0,l	retn

70	F	46	ld b,(hl)	bit 0,(hl)	im 0
71	G	47	ld b,a	bit 0,a	ld i,a
72	H	48	ld c,b	bit 1,b	in c,(c)
73	I	49	ld c,c	bit 1,c	out (c),c
74	J	4A	ld c,d	bit 1,d	adc hl,bc
75	K	4B	ld c,e	bit 1,e	ld bc,(NN)
76	L	4C	ld c,h	bit 1,h	
77	M	4D	ld c,l	bit 1,l	reti
78	N	4E	ld c,(hl)	bit 1,(hl)	
79	O	4F	ld c,a	bit 1,a	ld r,a
80	P	50	ld d,b	bit 2,b	in d,(c)
81	Q	51	ld d,c	bit 2,c	out (c),d
82	R	52	ld d,d	bit 2,d	sbc hl,de
83	S	53	ld d,e	bit 2,e	ld (NN),de
84	T	54	ld d,h	bit 2,h	
85	U	55	ld d,l	bit 2,l	
86	V	56	ld d,(hl)	bit 2,(hl)	im 1
87	W	57	ld d,a	bit 2,a	ld a,i
88	X	58	ld e,b	bit 3,b	in e,(c)
89	Y	59	ld e,c	bit 3,c	out (c),e
90	Z	5A	ld e,d	bit 3,d	adc hl,de
91	[	5B	ld e,e	bit 3,e	ld de,(NN)
92	/	5C	ld e,h	bit 3,h	
93	]	5D	ld e,l	bit 3,l	
94	↑	5E	ld e,(hl)	bit 3,(hl)	im 2
95	_	5F	ld e,a	bit 3,a	ld a,r
96	£	60	ld h,b	bit 4,b	in h,(c)
97	a	61	ld h,c	bit 4,c	out (c),h
98	b	62	ld h,d	bit 4,d	sbc hl,hl
99	c	63	ld h,e	bit 4,e	ld (NN),hl
100	d	64	ld h,h	bit 4,h	
101	e	65	ld h,l	bit 4,l	
102	f	66	ld h,(hl)	bit 4,(hl)	
103	g	67	ld h,a	bit 4,a	rrd
104	h	68	ld l,b	bit 5,b	in l,(c)
105	i	69	ld l,c	bit 5,c	out (c),l
106	j	6A	ld l,d	bit 5,d	adc hl,hl
107	k	6B	ld l,e	bit 5,e	ld hl,(NN)
108	l	6C	ld l,h	bit 5,h	
109	m	6D	ld l,l	bit 5,l	
110	n	6E	ld l,(hl)	bit 5,(hl)	
111	o	6F	ld l,a	bit 5,a	rld
112	p	70	ld (hl),b	bit 6,b	in f,(c)
113	q	71	ld (hl),c	bit 6,c	
114	r	72	ld (hl),d	bit 6,d	sbc hl,sp
115	s	73	ld (hl),e	bit 6,e	ld (NN),sp

116	t	74	ld (hl),h	bit 6,h	
117	u	75	ld (hl),l	bit 6,l	
118	v	76	halt	bit 6,(hl)	
119	w	77	ld (hl),a	bit 6,a	
120	x	78	ld a,b	bit 7,b	in a,(c)
121	y	79	ld a,c	bit 7,c	out (c),a
122	z	7A	ld a,d	bit 7,d	adc hl,sp
123	{	7B	ld a,e	bit 7,e	ld sp,(NN)
124	|	7C	ld a,h	bit 7,h	
125	}	7D	ld a,l	bit 7,l	
126	~	7E	ld a,(hl)	bit 7,(hl)	
127	©	7F	ld a,a	bit 7,a	
128	□	80	add a,b	res 0,b	
129	▝	81	add a,c	res 0,c	
130	▘	82	add a,d	res 0,d	
131	▀	83	add a,e	res 0,e	
132	▗	84	add a,h	res 0,h	
133	▐	85	add a,l	res 0,l	
134	▚	86	add a,(hl)	res 0,(hl)	
135	▜	87	add a,a	res 0,a	
136	▖	88	adc a,b	res 1,b	
137	▞	89	adc a,c	res 1,c	
138	▌	8A	adc a,d	res 1,d	
139	▛	8B	adc a,e	res 1,e	
140	▄	8C	adc a,h	res 1,h	
141	▟	8D	adc a,l	res 1,l	
142	▙	8E	adc a,(hl)	res 1,(hl)	
143	█	8F	adc a,a	res 1,a	

144	(a)		90	sub b	res 2,b	
145	(b)		91	sub c	res 2,c	
146	(c)		92	sub d	res 2,d	
147	(d)		93	sub e	res 2,e	
148	(e)		94	sub h	res 2,h	
149	(f)		95	sub l	res 2,l	
150	(g)		96	sub (hl)	res 2,(hl)	
151	(h)		97	sub a	res 2,a	
152	(i)		98	sbc a,b	res 3,b	
153	(j)	גרפיקה־	99	sbc a,c	res 3,c	
154	(k)	מוגדרת־	9A	sbc a,d	res 3,d	
155	(l)	המפעיל	9B	sbc a,e	res 3,e	
156	(m)		9C	sbc a,h	res 3,h	
157	(n)		9D	sbc a,l	res 3,l	
158	(o)		9E	sbc a,(hl)	res 3,(hl)	
159	(p)		9F	sbc a,a	res 3,a	
160	(q)		A0	and b	res 4,b	ldi
161	(r)		A1	and c	res 4,c	cpi
162	(s)		A2	and d	res 4,d	ini
163	(t)		A3	and e	res 4,e	outi
164	(u)		A4	and h	res 4,h	
165	**RND**		A5	and l	res 4,l	
166	**INKEY$**		A6	and (hl)	res 4,(hl)	
167	**PI**		A7	and a	res 4,a	
168	**FN**		A8	xor b	res 5,b	ldd
169	**POINT**		A9	xor c	res 5,c	cpd
170	**SCREEN$**		AA	xor d	res 5,d	ind
171	**ATTR**		AB	xor e	res 5,e	outd
172	**AT**		AC	xor h	res 5,h	
173	**TAB**		AD	xor l	res 5,l	
174	**VAL$**		AE	xor (hl)	res 5,(hl)	
175	**CODE**		AF	xor a	res 5,a	
176	**VAL**		B0	or b	res 6,b	ldir
177	**LEN**		B1	or c	res 6,c	cpir
178	**SIN**		B2	or d	res 6,d	inir
179	**COS**		B3	or e	res 6,e	otir
180	**TAN**		B4	or h	res 6,h	
181	**ASN**		B5	or l	res 6,l	
182	**ACS**		B6	or (hl)	res 6,(hl)	
183	**ATN**		B7	or a	res 6,a	
184	**LN**		B8	cp b	res 7,b	lddr
185	**EXP**		B9	cp c	res 7,c	cpdr
186	**INT**		BA	cp d	res 7,d	indr
187	**SQR**		BB	cp e	res 7,e	otdr
188	**SGN**		BC	cp h	res 7,h	
189	**ABS**		BD	cp l	res 7,l	

190	**PEEK**	BE	cp (hl)	res 7,(hl)
191	**IN**	BF	cp a	res 7,a
192	**USR**	C0	ret nz	set 0,b
193	**STR$**	C1	pop bc	set 0,c
194	**CHR$**	C2	jp nz,NN	set 0,d
195	**NOT**	C3	jp NN	set 0,e
196	**BIN**	C4	call nz,NN	set 0,h
197	**OR**	C5	push bc	set 0,l
198	**AND**	C6	add a,N	set 0,(hl)
199	<=	C7	rst 0	set 0,a
200	>=	C8	ret z	set 1,b
201	<>	C9	ret	set 1,c
202	**LINE**	CA	jp z,NN	set 1,d
203	**THEN**	CB		set 1,e
204	**TO**	CC	call z,NN	set 1,h
205	**STEP**	CD	call NN	set 1,l
206	**DEF FN**	CE	adc a,N	set 1,(hl)
207	**CAT**	CF	rst 8	set 1,a
208	**FORMAT**	D0	ret nc	set 2,b
209	**MOVE**	D1	pop de	set 2,c
210	**ERASE**	D2	jp nc,NN	set 2,d
211	**OPEN #**	D3	out (N),a	set 2,e
212	**CLOSE #**	D4	call nc,NN	set 2,h
213	**MERGE**	D5	push de	set 2,l
214	**VERIFY**	D6	sub N	set 2,(hl)
215	**BEEP**	D7	rst 16	set 2,a
216	**CIRCLE**	D8	ret c	set 3,b
217	**INK**	D9	exx	set 3,c
218	**PAPER**	DA	jp c,NN	set 3,d
219	**FLASH**	DB	in a,(N)	set 3,e
220	**BRIGHT**	DC	call c,NN	set 3,h
221	**INVERSE**	DD	prefixes instructions using ix	set 3,l
222	**OVER**	DE	sbc a,N	set 3,(hl)
223	**OUT**	DF	rst 24	set 3,a
224	**LPRINT**	E0	ret po	set 4,b
225	**LLIST**	E1	pop hl	set 4,c
226	**STOP**	E2	jp po,NN	set 4,d
227	**READ**	E3	ex (sp),hl	set 4,e
228	**DATA**	E4	call po,NN	set 4,h
229	**RESTORE**	E5	push hl	set 4,l
230	**NEW**	E6	and N	set 4,(hl)
231	**BORDER**	E7	rst 32	set 4,a
232	**CONTINUE**	E8	ret pe	set 5,b
233	**DIM**	E9	jp (hl)	set 5,c
234	**REM**	EA	jp pe,NN	set 5,d

235	**FOR**	EB	ex de,hl	set 5,e
236	**GO TO**	EC	call pe,NN	set 5,h
237	**GO SUB**	ED		set 5,l
238	**INPUT**	EE	xor N	set 5,(hl)
239	**LOAD**	EF	rst 4Ø	set 5,a
24Ø	**LIST**	FØ	ret p	set 6,b
241	**LET**	F1	pop af	set 6,c
242	**PAUSE**	F2	jp p,NN	set 6,d
243	**NEXT**	F3	di	set 6,e
244	**POKE**	F4	call p,NN	set 6,h
245	**PRINT**	F5	push af	set 6,l
246	**PLOT**	F6	or N	set 6,(hl)
247	**RUN**	F7	rst 48	set 6;a
248	**SAVE**	F8	ret m	set 7,b
249	**RANDOMIZE**	F9	ld sp,hl	set 7,c
25Ø	**IF**	FA	jp m,NN	set 7,d
251	**CLS**	FB	ei	set 7,e
252	**DRAW**	FC	call m,NN	set 7,h
253	**CLEAR**	FD	prefixes instruc- tions using iy	set 7,l
254	**RETURN**	FE	cp N	set 7;(hl)
255	**COPY**	FF	rst 56	set 7,a

נספח B : דווחים

הדווחים מופיעים בתחתית המסך בכל עת שהמחשב חדל מביצוע התכנית, ומסביר מדוע עצר, באם מסיבה טבעית או מאחר והתרחשה שגיאה.
הדווח מכיל קוד מספרי או אות, כך שתוכל להתייחס לטבלה שבהמשך, הודעה קצרה המסבירה מה התרחש ואת מספר השורה ומספר ההצהרה בתוך אותה שורה בה הופסק הביצוע. (פקודה מוצגת כשורה Ø. בתוך שורה, הצהרה מספר 1 היא ההצהרה הראשונה, הצהרה 2 היא זו שלאחר הנקודותיים (:) הראשונות או THEN וכך הלאה).
התנהגותה של CONTINUE תלויה בדווחים. באופן נורמלי, CONTINUE ממשיכה לשורה ולהצהרה שהוגדרו בדווח האחרון, אולם קיימים יוצאים מן הכלל, עם הדווחים Ø, 9 ו־D (ראה גם בנספח C).
לפנינו טבלה המציגה את כל הדווחים. היא מציינת גם באילו נסיבות הדווח יכול להתקבל; עליך להתייחס בנקודה זו — גם לנספח C. לדוגמא־הדווח: A Invalid Argvment יכול להתרחש עבור SQR, IN, ACS ו־ASN והגדרות נתוניהן — כפי שמופיעות בנספח C — יראו לך במה שגית.

קוד	**משמעות**	**הסיטואציה**
Ø	OK (בסדר) סיום מוצלח, או קפיצה למספר שורה הגדול מהקיים. דווח זה אינו משנה את השורה וההצהרה אליה התבצעה קפיצה על ידי CONTINUE.	כל־שהיא
1	NEXT without FOR (NEXT ללא FOR) משתנה הבקרה אינו קיים (הוא לא נקבע על ידי הצהרת FOR), אך קיים משתנה רגיל באותו שם.	NEXT
2	Variable not found (משתנה לא מוכר) עבור משתנה פשוט — דווח זה יתקבל באם יעשה שימוש במשתנה בטרם יוגדר על ידי LET, READ או INPUT או יוטען מטייפ או יוגדר בהצהרת FOR. עבור משתנה מותווה (subscripted) הדבר יתרחש באם יעשה שימוש־במשתנה בטרם הוגדרו מימדיו ע״י DIM או הוטען מהטייפ.	כל־שהיא
3	Subscript wrong (תוית שגויה) התוית היא מעבר למימד שהוגדר למערך, או שקיים מספר שונה של תויות ממספר המימדים. באם	משתנים מותווים תת־מחרוזות

התוית שלילית או גדולה מ־65535
אזי דווח שגיאה B יודפס.

4	Out of memory (לא נותר מקום בזכרון) אין מקום מספיק במחשב עבור מה שאתה מנסה לבצע. באם נראה כי המחשב "נתקע" במצב זה ללא יכולת להמשיך, יתכן ותצטרך לנקות את שורת הפקודה בעזרת DELETE ואחר להשמיט מהזכרון שורה או שתיים (תוך כוונה להחזירן ברגע הראשון־האפשרי) כדי להותיר לך מקום לתמרן. עם — נניח — CLEAR.	LET, INPUT, FOR, DIM, GO SUB, LOAD, MERGE. לעתים בעת חישוב־בטויים.
5	Out of screen (לא נותר מקום במסך) הצהרת INPUT נסתה ליצור יותר מ־23 שורות בחלק התחתון של המסך. כמו כן קורה כאשר מודפסת ההצהרה PRINT AT 22,...	INPUT, PRINT AT
6	Number too big (המספר גדול מדי) החשובים הולידו מספר הגדול־בקרוב מ־10^{38}	כל פעולה אריתמטית
7	RETURN without GO SUB (RETURN ללא GO SUB) קיים RETURN אחד יותר ממספר ה־GO SUB הקיימים.	RETURN
8	End of File. (קצה הקובץ)	פעולות הקשורות — בכונן הדיסקט, וכדומה
9	STOP הצהרה לאחריה, CONTINUE לא יחזור על ה־STOP, אלא ימשיך מההצהרה שלאחריה.	STOP
A	Invalid argument (ארגומנט לא חוקי) הארגומנט לפונקציה אינו חוקי מסיבה כלשהיא.	SQR, LN, ASN, ACS, USR (עם ארגומנט־מחרוזת)
B	Integer out of range (שלם גדול מדי) כאשר נדרש שלם, ארגומנט הנקודה הצפה מעוגל לשלם הקרוב ביותר. באם הוא מחוץ לתחום הקיים — מתקבלת שגיאה B.	RUN, RANDOMIZE, POKE, DIM, GO TO, GO SUB, LIST, LLIST, PAUSE, PLOT, CHR$, PEEK, USR (עם ארגומנט־מספרי)
	עבור גישה למערך ראה — פרק 3	גישה למערך

C	Nonsense in BASIC (שטויות ב־BASIC) התוכן של משתנה (המחרוזת) אינו יוצר בטוי בעל משמעות.	VAL, VHL$
D	BREAK — CONT repeats BREAK נלחץ בזמן־פעולה עם התקן הקפי. התנהגותה של CONTINUE לאחר דווח זה היא רגילה בכך שהיא חוזרת על ההצהרה. השווה עם דווח L.	LOAD, SAVE, VERIFY, MERGE, LPRINT, LLIST, COPY. כמו כן, כאשר המחשב שואל scroll? ותשובתך N, SPACE או STOP.
E	Out of DATA (לא נותר מידע) ניסית לבצע READ מעבר לקצה רשימת ה־DATA.	READ
F	Invalid File name (שם קובץ — לא חוקי) SAVE עם שם ריק או שם המכיל מעל 1Ø תוים.	SAVE
G	No room for line (אין מקום לשורה) לא נותר די מקום כדי לצרף את השורה החדשה לזכרון.	בעת הכנסת שורה לתכנית.
H	STOP in INPUT (STOP בתוך INPUT) המידע ל־INPUT התחיל עם STOP, או מקש תו המקף (—) עבור INPUT LINE — נלחץ.	INPUT
I	FOR without NEXT (FOR ללא NEXT) התקימה לולאת FOR שצריכה היתה להתבצע אף פעם (כלומר FOR n=1 TO Ø) והצהרת NXET המתאימה אינה ידועה.	FOR
J	Invalid I/O device (התקן קלט/פלט לא תקין)	פעולות כונן דיסקט וכדומה
K	Invalid colour (צבע לא חוקי) המספר שהוגדר אינו ערך — מתאים.	INK, PAPER, BORDER, FLASH, BRIGHT, INVERSE, OVER; כמו־כן לאחר אחד מתוי הבקרה.
L	BREAK into program (BREAK בתוך תכנית) הלחיצה על BREAK מפוענחת בין שתי הצהרות. השורה ומספר ההצהרה בדווח מציינים את	כלשהיא

השורה שלפני הלחיצה על BREAK, אולם CONTINUE ימשיך מההצהרה שלאחריה (ומאפשר קיום קפיצות), כך שאין חזרה על הצהרות.

M	RAMTOP no good (ערך RAMTOP לא תקין); המספר שהוגדר עבור RAMTOP הינו גדול או קטן מדי.	CLEAR; בדרך כלל ב־RUN.
N	Statement lost (ההצהרה אבדה) קפיצה להצהרה שאינה קימת יותר בזיכרון	RETURN, NEXT, CONTINUE
O	Invalid stream (רצף לא חוקי)	בפעולות כונן — הדיסקט וכו'.
P	FN withovt DEF (FN ללא DEF) פונקציה מוגדרת ע"י המשתמש.	FN
Q	Parameter error מספר שגוי של ארגומנטים, או שאחד מהם הינו מסוג לא מתאים (מחרוזת במקום מספר או להפך)	FN
R	Tape loading error (שגיאת הטענה מטייפ) הקובץ שהוגדר נמצא בטייפ, אך מסיבה כלשהיא לא נתן לקרוא אותו או שאינו מאומת.	VERIFY, LOAD או MERGE

נספח C: תיאור תמציתי של :ZX Spectrum

החלק הראשון של נספח זה חוזר על העובדות הנוגעות ללוח המקשים והמסך — אשר הוסברו בספרון — המבוא.

לוח המקשים:

התוים במחשב זה מכילים לא רק סימולים בודדים (אותיות, ספרות וכו') אלא גם צירופי־סימולים (כגון מילות־מקש, שמות־פונקציות ועוד) — כאשר כל אלו מוכנסים למחשב מלוח המקשים. כדי לקבל פונקציות ופקודות אלו — למספר מקשים חמש או יותר משמעויות שונות, המוגדרות בעזרת מקשי ה־SHIFT (CAPS SHIFT או — SYMBOL SHIFT), וזאת ע"י לחיצה ממושכת עליהם בעת שלוחצים על מקש אחר וע"י הפעלת המכונה באופני־פעולה (Modes) שונים.

אופן־הפעולה מצוין בעזרת הסמן (cursor) שהוא אות מהבהבת המראה היכן יודפס התו הבא שיוכנס מלוח־המקשים. אופן הפעולה K (מהמילה Keywords = 'מילות מקש') מחליף את אופן־הפעולה L בעת שהמכונה מצפה לפקודה או לשורת־תכנית (לא למידע מ־INPUT), וממקומו על השורה יודעת המערכת באם עליה לצפות למספר־שורה או למילת מקש. כך הדבר בתחילת שורה או לאחר THEN או לאחר: (מלבד במחרוזת). באם לא נעשה שימוש באחד ממקשי ה־SHIFT, המקש הבא יפורש או במילת־מקש (הכתובה על המקשים) או כספרה.

אופן־פעולה L (מהמילה Letters = 'אותיות') מתקיים בדרך כלל בכל שאר הזמן. באם לא נעשה שימוש במקשי ה־SHIFT, המקש הבא יפורש כסימול העיקרי שעל המקש — באותיות כתיב עבור אותיות. בשני אופני־הפעולה K ו־L, SYMBOL SHIFT עם מקש יפורש — כתו הצדדי האדום שעל המקש ו־CAPS SHIFT עם מקש־ספרה יפורש כפונקצית הבקרה הרשומה בלבן מעל המקש. ל־CAPS SHIFT עם מקשים אחרים אין כל השפעה על מילות המקש באופן הפעולה — K, ובאופן הפעולה L הוא גורם להפיכת אותיות כתוב לאותיות דפוס.

אופן־הפעולה C (מהמילה Capitals = "אותיות דפוס") הינו גרסה של אופן פעולה L, כשבו כל האותיות מופיעות כאותיות דפוס. המקש CAPS SHIFT גורם למעבר באופן־פעולה L ל־C או בכוון ההפוך.

באופן הפעולה E (מהמילה Extended = "מורחב") משמש להשגת תוים נוספים, בעקר צירופים. הוא מתקבל לאחר ששני מקשי ה־SHIFT נלחצו יחדיו וזאת למשך הקשה אחת על מקש כלשהוא. באופן פעולה זה, מקש־אות מגדיר תו אחד או צרוף (המוצג בירוק מעליו) באם לא נעשה שימוש במקשי ה־SHIFT, ואחר (כמוצג באדום מתחתיו) באם נלחץ עם אחד משני מקשי ה־SHIFT. מקש ספרה מגדיר צרוף באם נלחץ עם SYMBOL SHIFT; בכל מקרה אחר הוא מגדיר סדרה לבקרת הצבע.

אופן הפעולה G (מהמילה Graphics = 'גרפיקה') מתקבל לאחר הקשת GRAPHICS (CAPS SHIFT עם 9), ונמשך עד אשר נקיש זאת שוב. מקש־ספרה יגדיר גרפיקת־פסיפס, יפסיק את GRAPHICS או DELETE, וכל אחד ממקשי־האותיות מלבד Y, X, W, V ו־Z מציג גרפיקה — שאותה הגדיר המשתמש. באם מקש כלשהוא נלחץ באופן רציף מעל פרק זמן של 2 או 3 שניות, הוא יתחיל לחזור על עצמו. הקלט — מלוח המקשים מופיע במחצית התחתונה של המסך בעת שהוא מודפס, כאשר כל תו (סימול יחיד או צרוף סימולים) מוכנס לפני הסמן. נתן להזיז את הסמן שמאלה בעזרת CAPS SHIFT עם 5 או ימינה על ידי — CAPS SHIF עם 8. נתן להשמיט את התו שלפני הסמן בעזרת DELETE

(CAPS SHIFT עם ∅). (הערה: נתן להשמיט את כל השורה על ידי הדפסת EDIT (CAPS SHIFT עם 1) ולאחריו ENTER).

כאשר מקישים על ENTER השורה מבוצעת, מוכנסת לתכנית או משמשת כמידע ל־INPUT בהתאם, אלא אם היא מכילה שגיאת תחביר. במקרה זה ? יופיע בסמוך לשגיאה.

בעת ששורות התכנית מוכנסות למחשב, רישומן מופיע במחצית העליונה של המסך. האופן בו נוצר רישום זה מסובך קמעא והוסבר בפרק 2. השורה האחרונה שהוכנסה נקראת ה'שורה־הנוכחית' והיא מצוינת על ידי הסימול <, אולם נתן לשנות זאת בעזרת המקשים ⇩ (CAPS SHIFT עם 6) וְ־ ⇧ (CAPS SHIFT עם 7). באם EDIT (CAPS SHIFT עם 1) נלחץ, 'השורה הנוכחית' מובאת לתחתית המסך, שם היא נתנת לעריכה.

בעת שמבוצעת פקודה או תכנית "רצה", הפלט מוצג במחצית העליונה של המסך ונשאר עד אשר מוכנסת שורת תכנית, או ENTER נלחץ עם שורה ריקה, או המקש ⇧ או ⇩ נלחץ. בחלק התחתון יופיע דווח המספק קוד.(ספרה או אות) המפנה אותך לנספח B (לשגיאה המתאימה), תמצית מילולית של האמור בנספח B, מספר השורה המכילה את ההצהרה האחרונה שבוצעה (או ∅ עבור פקודה) ומקומה של ההצהרה בתוך השורה. הדווח נשאר על המסך עד אשר נלחץ מקש (ומציין בכך — אופן פעולה K).

בנסיבות מסוימות, CAPS SHIFT עם המקש SPACE מתנהג כמו BREAK, בעצרו את המחשב בדווח D או L. הדבר נתן להבחנה —

(i) בסוף הצהרה בעת שהתכנית "רצה", או

(ii) בעת שהמחשב משתמש בטייפ־הקסיטות או במדפסת.

מסך הטלויזיה:

המסך בנוי מ־24 שורות בנות 32 עמודות כל אחת והוא מחולק לשני חלקים. החלק העליון הוא לכל היותר בן 22 שורות ומציג או רישום (listing) או פלט־מתכנית. בעת שההדפסה בחלק העליון הגיעה לתחתית, המסך "מתגלגל" (Scrolls) שורה אחת כלפי מעלה; באם בכך יגרם איבוד שורה שאותה עדיין לא הספקת לראות, המחשב יבקש רשותך לגלגל את המסך בשאלה — ?Scroll. הקשה על N, SPACE או STOP יגרמו לתכנית לעצור עם הדווח: D BREAK — CONT repeats; כל שאר המקשים יאפשרו את המשכת הגלגול. החלק התחתון משמש להכנסת פקודות, שורות תכנית ומידע ל־INPUT וכן — להצגת דווחים. החלק התחתון מתחיל בגודל שתי שורות (העליונה בהם — ריקה) אך הוא מתרחב כדי לשמור את כל שמודפס למחשב בעת שהוא מגיע למיקום ה־PRINT הנוכחי בחלק העליון, הרחבות נוספות יביאו את המחצית העליונה להתגלגל גם כן — כלפי מעלה.

לכל מיקום־תו התכונות המגדירות את צבעי ה'ניר' (החזית) וה'דיו' (הרקע), שתי דרגות בהירות וקיום הבהוב.

הצבעים האפשריים הנם שחור, כחול, אדום, סגול, ירוק, תכול, צהוב ולבן.

קצה המסך נתן לקביעה לכל אחד מצבעים בעזרת הצהרת BORDER.

כל מיקום תו מחולק ל־8×8 נקודות וגרפיקה־בהבחנה גבוהה מתקבלת ע"י הגדרת הנקודות הללו בצורה אינדיוידואלית באם יציגו את צבע ה'ניר' או צבע ה'דיו' — עבור אותו מיקום תו.

התכונות במיקום־התו מכוונת בכל עת שתו נכתב, או בעת שנקודה משורטטת שם. הצורה המדויקת של הכוון נקבעת ע"י פרמטרי ה־PRINT — מהן קיימות שתי קבוצות (הקרויות — קבועה וזמנית) של שש: הפרמטרים PAPER, INK, FLASH, BRIGHT, INVERSE וְ־OVER. פרמטרים קבועים לחלק העליון נקבעים על־ידי הצהרות PAPER, INK וכו', ונמשכים עד לפניה הבאה. (במצב ההתחלתי, עם הדלקת המערכת, הם מוגדרים כ'דיו' שחורה על 'ניר' לבן, בהירות רגילה, ללא הבהוב, תצוגת וידיאו רגילה וללא כתיבה חוזרת). פרמטרים קבועים עבור החלק התחתון משתמשים בצבע ה'גבול' כצבע ה'ניר' בניגוד גוונים שחור או לבן עבור ה'דיו', בהירות רגילה, ללא הבהוב, וידיאו רגיל וללא כתיבה חוזרת.

פרמטרים זמניים מוגדרים על ידי פרטי PRINT, INK וכו' אשר שובצו בהצהרות PRINT, LPRINT, INPUT, PLOT, DRAW וְ־CIRCLE, וכמו כן עלידי תוי הבקרה של PAPER, INK וכו' בעת שהם מודפסים

לטלויזיה — הם מלווים בבית נוסף המגדיר את ערך הפרמטר. פרמטרים זמניים נמשכים רק עד לסוף הצהרת ה־PRINT (או מה שלא תהיה), או בהצהרת INPUT, עד אשר נדרש מידע עבור ה־INPUT מלוח המקשים, כאשר הם מוחלפים בפרמטרים הקבועים.

פרמטרים INK PAPER הם בתחום ∅ עד 9. הפרמטרים ∅ עד 7 הם הצבעים הנמצאים בשימוש בעת הדפסת תו:

0 שחור
1 כחול
2 אדום
3 סגול
4 ירוק
5 תכול
6 צהוב
7 לבן

הפרמטר 8 מציין באם הצבע במסך ישאר ללא שנוי בעת הדפסת תו.

הפרמטר 9 מציין באם הצבע הנדון ('נייר' או 'דיו') יהיה לבן או שחור כדי להראות ברקע הצבע השני.

פרמטרי FLASH וְ־BRIGHT הם ∅ וְ־1, או 8: הָ־1 משמעותו שתהיה בהירות גבוהה או הבהוב יתחיל, וְ־8 משמעותו שהמצב הקודם ישאר ללא שנוי.

פרמטרי OVER וְ־INVERSE הם ∅ וְ־1:

∅ OVER — תוים חדשים מוחקים את התוים הישנים.

1 OVER — תבנית הביטים הישנה והתוים החדשים מצורפים יחד בעזרת הפעולה הלוגית "או־בלעדי" (Exclusive OR). תהליך זה נקרא כתיבה־מחדש (overprinting).

∅ INVERSE — התוים החדשים מודפסים כצבע ה'דיו' על צבע ה'ניר' ('וידיאו רגיל').

1 INVERSE — התוים החדשים מודפסים כצבע ה'ניר' על צבע ה'דיו' ('וידיאו — הפוך').

כאשר תו הבקרה TAB מתקבל על ידי הטלויזיה, שני בתים נוספים אמורים להתלוות אליו — בכדי לקבוע את מיקום ה־TAB — n (הבית המשמעותי פחות — ראשון). זה מופחת מודולו 32 לְ־No (הבה נניח זאת), ואז מרווחים מספיקים מודפסים כדי להניע את מיקום ההדפסה לעמודה No.

כאשר תו הבקרה — פסיק מתקבל, אזי מודפסים מספר מרווחים (לפחות) כדי להניע את מיקום ההדפסה לעמודה ∅ או עמודה 16.

כאשר מתקבל תו־הבקרה של ENTER, מיקום ההדפסה מועבר לשורה הבאה.

המדפסת:

הפלט ממחשב זה למדפסת — מתקבל באמצעות חוצץ (buffer) באורך שורה אחת (32 תוים), ושורה נשלחת אל המדפסת —

(i) :אשר ההדפסה ממלאת שורה

(ii) כאשר התקבל תו ENTER

(iii) בסוף התכנית, באם נותר דבר מה שלא הודפס.

(iv) כאשר בקרת — TAB ובקרת־פסוק מניעים את מקום ההדפסה לשורה חדשה.

בקרות הָ־TAB ובקרות הפסיק קובעות מרווחים בצורה דומה — למה שהתבצע בטלויזיה.

בקרת AT משנה את מיקום ההדפסה בעזרת מספר העמודה ומתעלמת ממספר השורה.

ההדפסה מושפעת מבקרות INVERSE וְ־OVER (וכמו כן הצהרות) בצורה דומה למסך, אך לא על ידי PAPER, INK, FLASH, או BRIGHT.

המדפסת תעצור עם דווח B באם נלחץ BREAK.

באם המדפסת אינה מחוברת — המידע פשוט יאבד.

:BASIC

המספרים נשמרים בדיוק של 9 עד 1∅ ספרות. המספר הגדול ביותר הנתן לאחסון הוא בקירוב 10^{38}, והקטן ביותר (החיובי) הוא בקירוב $4*10^{-39}$.

המספרים נשמרים בָּ־ZX Spectrum במספרים בינארים בנקודה צפה, עם בית אקספוננט אחד — e‏ (1<=e<=255) וארבעה בתי מנטיסה — m ‏(½< = m<1).

התבנית מייצגת את המספר —

$$m*e^{-128}$$

מאחר וְ ½< = m<1, הביט המשמעותי ביותר במנטיסה הוא תמיד 1. לכן למעשה נתן להחליפו בביט שיציג את הסימן: ∅ עבור מספר חיובי וְ־1 עבור שלילי.

לשלמים־קטנים יצוג מיוחד שבו הבית הראשון הוא ∅, השני הוא ביט הסימן (∅ או FF h) והשלישי והרביעי הם השלמים בשיטת המשלים ל־2, כשהספרה הפחות משמעית־ראשונה.

למשתנים מספריים שמות באורך שונה, כשהתו הראשון חייב להיות אות. כל יתר התוים אינם מענינים. בקרות הצבע והמרווחים לא נלקחות בחשבון וכל האותיות מודפסות כאותיות כתיב.

למשתני הבקרה עבור לולאות FOR-NEXT שמות שאורכן אות אחת־ויחידה.

למערכים־מספריים שמות באורך אות יחידה, והוא יכול להיות זהה לשמו של משתנה פשוט. הם יכולים לכלול מספר שרירותי של מימדים וגודל־מימד שרירותי אף הוא. התוים מתחילים בְּ־1.

מחרוזות הנן גמישות לחלוטין באורכן. שם־המחרוזת מכיל אות יחידה שלאחריה $.

מערכי מחרוזות יכולים להיות בעל מספר מימדים כלשהוא — כשלכל מימד גודל שרירותי, השם הוא אות יחידה מלווה בסימול $ — והוא יכול שלא להיות זהה לשם המחרוזת.

כל המחרוזות במערך נתון הן בעלות אורך קבוע זהה לכולן, המוגדר כמימד נוסף בהצהרת DIM. התויות מתחילות בְּ־1.

פריסת מחרוזות (Slicing): תת־מחרוזות של מחרוזות נתנות לקביעה בעזרת "פורסים". פורס יכול להיות:

(i) ריק, או
(ii) בטוי מספרי, או
(iii) בטוי מספרי אפשרי To בטוי מספרי אפשרי

והוא משתמש לבטא תת־מחרוזות בעזרת:
(a) (פורס) בטוי מחרוזות
(b) (פורס, תוית ,..., תוית) משתנה־במערך מחרוזות

אשר משמעותו היא בדיוק כמו של —
(פורס) (תוית, ..., תוית) משתנה של מערך מחרוזות בְּ־(a) — הבה נניח כי בלטוי המחרוזות קיים הערך s$. באם הפורס הינו ריק, התוצאה היא s$ — הנחשבת כתת מחרוזות של עצמה.

באם הפורס הוא בטוי מספרי שערכו — m, אזי התוצאה היא התו הָ־'m' של s$ (תת־מחרוזות באורך 1) באם לפורס צורה כְּ־ (iii), אזי הבה נניח כי הבטוי המספרי הראשון הוא בעל הערך m (ברירת המחדל — 1), ולשני — n (ברירת המחדל היא אורכה של s$).

באם אורך s$=> n=> m = > 1, התוצאה היא תת־המחרוזות של s$ המתחילה בתו הָ־'m' ומסתיימת בתו הָ־n+.

באם >n=>0 התוצאה תהיה ה'מחרוזת הריקה'.

בכל מקרה אחר, יתקבל דווח שגיאה 3.

פריסת מחרוזות מתבצעת לפני ביצוע פונקציות או פעולות, אלא אם כן סוגרים מכתיבות סדר אחר.

תת־מחרוזות נתנת להשמה (assignment). ראה את האמור לגבי LET.

באם מרכאות אמורות להופיע בתוך ציטטה במחרוזות, יש לרשמן פעמיים.

:פונקציות

הארגומנט של פונקציה אינו דורש סוגרים באם הוא קבוע או משתנה (מותווה או פרוס).

פונקציה	סוג הארגומנט (x)	תוצאה:
ABS	מספר	ערך מוחלט
ACS	מספר	ארקסינוס ברדיאנים שגיאה A באם x לא בתחום: 1 — עד 1+.
AND	פעולה בינארית, כשהאופרנד הימני תמיד מספר. כשהאופנרד שמאלי — מספרי: כשהאופרנד השמאלי — מחרוזות: A$ B = ∅	A **AND** B = { A if B<>0 ; 0 if B=0 } A$ **AND** B = { A$ if B<>0 ; "" if B=0 }
ASN	מספר	ארקסינוס ברדיאנים. שגיאה A באם x לא בתחום 1 — עד 1+.
ATN	מספר	ארקטנגנס ברדיאנים.
ATTR	שני ארגומנטים — x וְ־y, שניהם מספרים — כשהם בתוך סוגריים.	המספר שבצורתו הבינארית יוצר קוד לתכונות שורה x (ועמודה y בטלויזיה. ביט 7 (המשמעותי ביותר) הוא 1 להבהוב, ∅ ליציב. ביט 6 הוא 1 לבהירות גבוהה וְ־∅ לרגילה. ביטים 5 עד 3 הם צבע ה'ניר' וביטים 2 עד ∅ הם צבע ה'דיו'.
BIN		שגיאה B אלא אם כן 23 = > x > = ∅ וכן 31 = > y > = ∅. זוהי למעשה לא פונקציה כי אם שיטת רישום מספרים: BIN שלאחריה 1 וְ∅ היא המספר ביצוג בינארי כזה.
CHR$	מספר	התו בעל הקוד המתאים לְ־x, כשהוא מעוגל לשלם הקרוב ביותר.
CODE	מחרוזות	הקוד של התו הראשון ב־x (או ∅ באם x הוא ה'מחרוזת הריקה')

COS	מספר (ברדיאנים)	קוסינוס x
EXP	מספר	e^x
FN		FN שלאחריה אות קוראת לפונקציה שהוגדרה ע״י המשתמש (ראה — DEF). הארגומנט חייב להופיע בסוגרים; גם אם אין כל ארגומנטים — הסוגרים חייבות להופיע.
IN	מספר	תוצאת הקלט מפורט x (port) (Ø = x < = FFFF h) (טוענת את צמד הרגיסטרים bc ב־x ומבצעת את ההוראה ב־a(c).)
INKEY$	אין	קוראת את לוח המקשים. התוצאה היא התו המייצג את המקש שנלחץ (באופני פעולה L או C באם נלחץ אחד בדיוק; אחרת — הקבוצה הריקה.
INT	מספר	החלק השלם (תמיד מעגלת כלפי מטה).
LEN	מחרוזת	אורך המחרוזת
LN	מספר	לוגריתם טבעי (בסיס e) שגיאה A עבור x< = o.
NOT	מספר	Ø : X <> o NOT x = 1 : X = Ø ל־NOT עדיפות 4.
OR	מספר פעולה בינארית כששני האופרנדים הם מספרים	1 : b <> Ø a OR b = Ø : b = Ø ל־OR עדיפות 2.
PEEK	מספר	ערך הבית בזכרון שכתובתו היא x (מעוגלת לשלם הקרוב ביותר). שגיאה B עבור x באם אינו בתום Ø עד 65535.
PI	אין	II (3.14159265...)
POINT	שני ארגומנטים, x ו־y	1 באם הנקודה (x,y) היא בצבע

	שהם מספרים; הם מופיעים בתוך סוגרים.	הדיו' ו־Ø עבור צבע 'הניר'. שגיאה B באם לא מתקיים O< = x < = 255 וכן Ø< = y< = 175.
RND	אין	המספר הפסאודו־אקראי הבא מיוצר ע״י לקיחת חזקות 75 מודולו 65537, החסרת 1 וחלוקה ב־65536. o< y < 1
SCREEN$	שני ארגומנטים — x ו־y, שניהם מספרים; הם מופיעים בסוגרים.	התו המופיע, רגיל או הפוך בטלויזיה בשורה X, עמודה Y. מספקת את ה'מחרוזת הריקה' באם התו אינו מוכר. שגיאה B אלא־אם מתקיים: Ø< = x < = 23 ו־Ø<=y< = 31
SGN	מספר	פונקצית הסימן (1— עבור שלילי, Ø עבור אפס ו־1+ עבור חיובי) של x.
SIN	מספר (ברדיאנים)	סינוס — x
SQR	מספר	השורש הרבועי. שגיאה A באם x<O.
STR$	מספר	מחרוזת התוים שתוצג באם x היה מודפס.
TAN	מספר (ברדיאנים)	טנגנס x
USR	מחרוזת	כתובת תבנית־הביטים עבור הסימול הגרפי שהוגדר ע״י המשתמש, המתאים ל־x. שגיאה A באם x אינו אות יחידה בין a ל־U או גרפיקה שהוגדרה ע״י המשתמש.
VAL	מחרוזת	מחשבת את x (ללא המרכאות שבקצותיו) כבטוי מספרי, שגיאה C באם x מכיל שגיאות תחביר או נותן ערך מחרוזת. שגיאות אחרות גם הן אפשריות, בתלות בבטוי.

ומקום שם את המחסנית הַ־GOSUB החדשה.

CLOSE — אינה פועלת ללא כונן־הדיסק וכו׳.

CLS — (Clear Screen)"מנקה" את קובץ התצוגה.

CONTINUE — ממשיכה את התכנית, החל מהמקום בו נעזבה בעת שעצרה בפעם האחרונה עם דווח שונה מְ־Ø. באם הדווח היה 9 או L, CONTINUE תמשיך מההצהרה הבאה בתור (תוך לקיחת קפיצות־בחשבון); אחרת היא חוזרת על הפקודה שבה התרחשה שגיאה. באם הדווח האחרון הופיע בשורת פקודה — אזי CONTINUE תנסה להמשיך את שורת הפקודה כשהיא תמשיך לתוך לולאה באם השגיאה היתה Ø:1, תדווח Ø באם היא היתה Ø:2 או שגיאה N באם היא היתה Ø:3 או גדולה יותר. CONTINUE מופיע כְ־CONT על לוח המקשים.

COPY — שולחת העתק של 22 השורות העליונות במסך למדפסת, באם מחוברת; בכל מצב אחר — אינה משפיעה. שים לב כי COPY אינה יכולה לשמש בהדפסה ב'רישום־אוטומטי' (automatic listing) המופיע על המסך. דווח D באם BREAK נלחץ.

הצהרות:

ברשימה זו —

- α — מייצגת אות יחידה
- v — מייצגת משתנה
- x, y, Z — מייצגים בטויים־מספריים
- m, n — מייצגים בטויים־מספרים שעוגלו לשלם הקרוב ביותר.
- e — מייצג בטוי.
- f — מייצג בטוי־מחרוזת.
- s — מייצג סדרת הצהרות המופרדות באמצעות נקודותיים (:).
- c — מייצג סדרת פרטי־צבע כשכל אחד מסתיים בפסיק או בפסיק־נקודה; לפרט צבע צורת ההצהרה הבאה — PAPER, INK, FLASH, BRIGHT, INVERSE או OVER.

שים לב כי בטויים שרירותיים מותרים בכל מקום (מלבד במקום מספר־השורה שבתחילת הצהרה).

כל ההצהרות מלבד INPUT, DEF ו־DATA נתנות לשימוש הן כפקודות והן בשורות בתוכנית (למרות שהן יותר מוחשיות באחד יותר מן השני). פקודה או שורת תכנית יכולה להכיל מספר הצהרות, המופרדות בעזרת נקודותיים (:). אין כל הגבלה היכן בשורה תופיע הצהרה מסוימת — למרות זאת ראה לגבי IF ו־REM.

BEEP x, y — משמיעה תו ברמקול למשך x שניות בגובה צלילי — y חצאי־טונים מעל C המרכזי (או מתחתיו באם y שלילי).

BORDER m — קובעת את צבע הגבול של המסך וכמו כן את צבע ה'ניר' עבור החלק התחתון של המסך. שגיאה K באם m לא בתחום Ø עד 7.

BRIGHT n — קובעת את בהירות התוים: n = Ø עבור בהירות רגילה, n = 1 לבהירות גבוהה ו־n = 8 עבור שקיפות (המצב הקודם). שגיאה K באם n אינו Ø, 1 או 8.

CAT — אינה פועלת ללא כונן־דיסק וכו׳

CIRCLE x, y, z — משרטט קשת־מעגל שמרכזו (x, y) ורדיוסו Z.

CLEAR — מוחק את כל המשתנים, בשחרורו על ידי כך את המקום שתפסו בזכרון. מבצע RESTORE ו־CLS, מאפס את מיקום הַ־PLOT לפינה השמאלית הנמוכה ו"מנקה" את מחסנית הַ־GO SUB.

CLEAR n — בדומה ל־CLEAR, אך באם אפשרי משנה את משתנה המערכת RAMTOP לערך n

VAL$	מחרוזת	מחשבת את x (ללא המרכאות בקצותיו) כבטוי מחרוזת. שגיאה C באם x מכיל שגיאת תחביר. שגיאות אחרות גם הן אפשריות כמו עבור VAL.
—	מספר	הפיכת המספר לשלילי.

הבאות — הנן פעולות בינאריות:

+	חיבור (עבור מספרים) או קשירת שרשרות (עבור מחרוזות).	
—	חיסור	
*	כפל	
/	חילוק	
↑	העלאה בחזקה. שגיאה B באם האופרנד השמאלי־שלילי.	
=	שווה	
<	גדול מ־ > קטן מְ	שני האופרנדים חייבים להיות מאותו סוגי התוצאה היא המספר 1 באם ההשוואה מתקיימת ו־Ø באם לא.
<=	קטן מְ־או שווה לו.	
>=	גדול מְ או שווה לו.	
<>	לא שווה.	

לפונקציות ולפעולות קיים סדר העדיפות הבא:

העדיפות	הפעולות
12	סימון בתויות ופריסה לתת־קבוצות.
11	כל הפונקציות מלבד NOT וסימן המינוס (להפיכת מספר לשלילי)
$\uparrow$ 10	
9	סימן המינוס (להפיכת מספר לשלילי)
8	*, /
6	+, — (מינוס כסימן חיסור)
5	=, >, <, <=, >=, <>
4	NOT
3	AND
2	OR

DATA e1 ,e2 ,e3 ,...	חלק מרשימת הָ־DATA. חייבת להופיע בתכנית.
DEF FN (1 , 2 ,... k) = e	אופן הגדרת פונקציה־המוגדרת־ע"י־המשתמש; חייבת להופיע בתכנית. כל אחד מְ־α וְ־1 α עד kα חייב להיות אות יחידה שלאחריה הסימול $ עבור תוצאה או־ארגומנט מחרוזת. מקבלת את הצורה DEF FN ()=e באם אין כל ארגומנטים.
DELETE f	אינה פועלת ללא כונן־הדיסקט וכו'
DIM (n1 ,... ,n_k)	מוחקת כל מערך קיים בשם ומגדירה מערך־מספרי בשם מ־בעל k מימדיםα,n_k, n1. מאתחלת את כל הערכים בו כְּ־Ø.
DIM $ (n1 ,... ,n_k)	מוחקת כל מערך או מחרוזת בשם $\alpha$$ ומגדירה מערך תוים בעל k מימדים n_k n1. מאתחלת את כל הערכים לְ"". נתן לראות זאת כמערך של מחרוזות באורך קבוע n_k-1 — בעל k-1 מימדים n1, ..., nk-1. שגיאה 4 מתקיימת באם אין די מקום עבור המערך בזכרון. מערך אינו מוגדר עד אשר הוגדרו מימדיו בהצהרת DIM.
DRAW x ,y	מקבילה לְ־DRAW x ,y ,Ø
DRAW x ,y ,z	משרטטת קו ממיקום ה־PLOT הנוכחי על ידי תנועה x אופקית וְ־y אנכית יחסית לה, בסיבוב בזוית Z שגיאה B באם הקו ממשיך מחוץ לגבולות המסך.
ERASE	אינה פועלת ללא כונן־הדיסקט וכו'.
FLASH n	מגדירה באם התוים יהבהבו או יראו יציבים. n=Ø עבור יציבים n = 1 למהבהבים וְ־n=8 למצב "שקוף" דהיינו מה שהיה בעבר.
FORα=x TO y	מקבילה לְ־1 FOR α =x TO y STEP.
FOR =x TO y STEP Z	מוחקת כל משתנה פשוט בשם α וקובעת משתנה בקרה בערך x, גבול y וגודל צעד z וכתובת לולאה המתייחסת להצהרה שלאחר הצהרת FOR. בודקת באם הערך ההתחלתי גדול (באם Ø< = STEP) או קטן (באם STEP < Ø) מאשר הגבול, ובאם כן מדלגת להצהרה NEXT α, כשהיא מדווחת על שגיאה 1 באם לא קיימת כזו. ראה בערך NEXT. שגיאה 4 מתקבלת באם אין מקום עבור משתנה הבקרה.
FORMAT f	אינה פועלת ללא כונן־הדיסקט וכו'
GOSUB n	דוחפת את מספר השורה של הצהרת GOSUB למחסנית; לאחר מכן מתנהגת כמו — GO TO n.
GO TO n	קופצת לשורה n (או, באם אינה קיימת, השורה הראשונה שלאחריה).

IF x THEN s	באם x 'נכון' (שונה מְ־Ø) — מבוצעת s. שים לב כי s היא כל ההצהרות עד לסוף השורה. הצורה 'מס' שורה IF x THEN' אינה מותרת.
INK n	קובעת את צבע ה'דיו' של התוים המודפסים. n היא בתחום Ø עד 7 עבור צבע, 8 עבור "שקיפות" (המצב הקודם) וְ־9 לניגודי־גוון. ראה גם נספח B. שגיאה K באם n אינו בתחום Ø עד 9.
INPUT...	הָ־'...' הינה סדרה של פרטי INPUT, מופרדים כמו בהצהרת PRINT בעזרת פסיק, פסיק־נקודה או גרשיים. פרט INPUT יכול להיות (i) כל פרט PRINT שאינו מתחיל באות (ii) שם משתנה, או (iii) LINE ולאחריה שם משתנה סוג המחרוזת. פרטי הָ־PRINT ומפרידיו שְׁב־(i) נחשבים בדיוק כבהצהרת PRINT, מלבד העובדה שכל שמודפס מופיע בתחתית המסך. עבור (ii) המחשב עוצר ומצפה לקלט של בטוי מִלוח המקשים; ערכו מושם למשתנה. הקלט מוחזר "כהד" — באופן הרגיל כאשר שגיאות תחביר גורמת להצגת ? מהבהב. לבטויים מסוג מחרוזות, בחוצץ הקלט מוגדרות במצב ההתחלתי שתי מרכאות־מחרוזת (הנתנות למחיקה באם הדבר נדרש). באם התו הראשון בקלט הוא STOP, התכנית עוצרת עם שגיאה H. (iii) דומה לְ־(ii) מלבד העובדה שהקלט נחשב כמחרוזת ללא מרכאות והמנגנון של הָ־STOP אינו פועל בו; כדי לעצור אתה חייב להדפיס ↓
INVERSE n	מביאה להיפוך התוים המודפסים. באם n = Ø התוים יודפסו ב'וידיאו רגיל' כצבע 'דיו' על צבע ה'ניר'. באם n = 1 התוים מודפסים ב'וידיאו הפוך', כלומר: ה'ניר' על צבע 'הדיו'. ראה גם בנספח B. שגיאה K באם n אינו Ø או 1.
LET v=e	מציבה את הערך e במשתנה v. LET אינו נתן להשמטה. משתנה פשוט הינו בלתי מוגדר עד שלא ייושם ע"י הצהרת LET, READ או INPUT. באם v הינו משתנה־מחרוזת־מותווה או משתנה בפרוסת מחרוזת (תת מחרוזת), אזי ההשמה היא

	בצורת "מיטת סדום" דהיינו באורך קבוע: משתנה המחרוזת של e או שהוא מקוצץ או שהוא מתמלא במרווחים מצידו הימני כדי להתאימו לאורך המשתנה v.
LIST	מקבילה לְ־LIST Ø
LIST n	מציגה את רישום התכנית בחלק העליון של המסך, החל מהשורה שמספרה הינו לפחות n — והופכת את n לשורה הנוכחית.
LLIST	מקבילה לְ־LLIST Ø
LLIST n	כמו LIST, אלא שעבור המדפסת.
LOAD f	מטעינה תכנית ומשתנים.
LOAD f DATA ()	מטעינה מערך מספרי.
LOAD f DATA $()	מטעינה מערך התוים $.
LOAD f CODE m, n	מטעינה לכל היותר n בתים החל מכתובת m.
LOAD f CODE m	מטעינה בתים החל מכתובת m.
LOAD f CODE	מטעינה בתים חזרה לכתובת בה נשמר קודם.
LOAD f SCREEN $	מקבילה לְ־LOAD f CODE 16384, 6912. מחפשת את הקובץ מהסוג המתאים בטייפ הקסיטות ומטעינה אותו תוך מחיקת הגרסאות הקודמות שבזכרון. ראה פרק 20.
LPRINT	בדומה לְ־PRINT אולם עבור המדפסת.
MERGE f	בדומה לְ־ LOAD f, אך אינה מוחקת שורות תכנית קודמת ומשתניה מלבד אלו שלהם אותו שם או מספר־שורה.
MOVE f1, f2	אינה פועלת ללא כונן־הדיסקט וכו'.
NEW	מתחילה את מערכת הָ־BASIC, תוך מחיקת תוכניות ומשתנים קודמים,

תוך שימוש בזכרון עד לבית שכתובתו
היא משתנה המערכת RAMBOT,
ושומרת על משתני המערכת UDG,
P RAMT, RASP ו־PIP.

NEXT α

(i) מוצאת את משתנה הבקרה α
(ii) מחבר את הצעד לערך
(iii) באם הצעד = > Ø וכן הערך > הגבול, או
הצעד <Ø וכן הערך < הגבול — היא
קופצת להצהרת־הלולאה.
שגיאה 2 — באם אין כל משתנה α.
שגיאה 1 — באם קיים אך אינו משתנה בקרה.

OPEN

אינה פועלת ללא כונן־הדיסקט וכו׳.

OUT m ,n

מוציאה בית n כפלט דרך פורט (port) m,
ברמת המיקרופרוססור. (מטעינה את צמד
הרגיסטרים bc ב־m, את הרגיסטר a
ב־n ומבצעת את הוראת האסמבלי: out (c), a)
Ø < = m < = 65535, —255 < = n < = 255, אחרת
מתקבלת שגיאה B.

OVER n

מבקרת את ההדפסה־החוזרת עבור התוים
בהדפסה.
באם n = Ø, התוים מוחקים את
הקודמים באותו מיקום.
באם n = 1 התוים החדשים מעורבבים
עם התוים הקודמים כדי ליצור צבע ׳דיו׳
בכל מקום בו אחד מהם היה בצבע
ה׳דיו׳ (אך לא שניהם) וצבע ׳ניר׳
באם שניהם היו בו צבע ׳ניר׳ או צבע
׳דיו׳. ראה גם בנספח B.
שגיאה K באם n אינו Ø או 1.

PAPER n

בדומה ל־INK, אלא שמבקרת
על צבע ה׳ניר׳ (הרקע).

PAUSE n

עוצרת את החישובים ומציגה את
קובץ התצוגה n מסגרות זמן
(5Ø מסגרות בשניה באירופה
ו־6Ø בצפון אמריקה) או עד
אשר נלחץ מקש כלשהוא.
Ø<=n<=65535 אחרת מתקבלת
שגיאה B. אם n = Ø, אזי PAUSE
אינה מוגבלת בזמן — אלא, נמשכת עד
ללחיצה על מקש כלשהוא.

PLOT c ,m ,n

מדפיסה נקודת ׳דיו׳ (כתלות
ב־OVER (INVERSE) ב׳מרכיב־התמונה׳
(pixel) — (|m|, |m|); מזיזה
את מיקום ה־PLOT.
אלא אם פרטי ה־C PRINT
מגדירים אחרת, צבע ה׳דיו׳ במיקום —
התו המכיל את ׳מרכיב־התמונה׳
משונה לצבע ה׳דיו׳ הנוכחי־הקבוע,
והשאר (צבע ה׳ניר׳, ההבהוב (בהירות)
נשארים ללא שינוי.
Ø<=|m|<= 255, Ø<=|n|<=175, אחרת
מתקבלת שגיאה B.

POKE m ,n

רושמת את הערך n בבית
שבכתובת m.
Ø< = m<= 65535, —255 <=n < = 255,
אחרת מתקבלת שגיאה B.

PRINT...

ה־׳...׳ הינה סדרה של פרטי
PRINT, המופרדים בפסיק, פסיק־
נקודה או גרשיים. הם נכתבים
בקובץ התצוגה כדי להשלח לטלויזיה.
לנקודה־פסוק (;) בין שני
פרטים אין כל השפעה: היא משמשת
פשוט להפרדה בין הפרטים. פסיק (,)
מוציא את תו־בקרת־הפסוק והגרש (׳)
מוציא את תו ה־ENTER.
בסופה של הצהרת PRINT, באם
אינה מסתיימת בנקודה־פסיק, פסיק
או גרש — מוצא כפלט התו ENTER.
פרט PRINT יכול להיות:
(i) ריק כלומר — כלום.
(ii) בטוי מספרי.
תחילה מודפס סימן מינוס באם הערך הוא
שלילי. כעת הבה נניח כי x הוא
המודולו של המספר.
באם $x<10^{-5}$ או $x>10^{13}$, המספר
יודפס בצורת רישום מדעית. המנטיסה
כוללת עד 8 ספרות (ללא ״זנב״ אפסים),
והנקודה העשרונית (לא קיימת עבור ספרה
אחת בלבד) היא לאחר הראשונה.
האקספוננט הוא E שלאחריו + או —,
שלאחריו ספרה או שתיים.
בכל מקרה אחר, x מודפס ברישום
הרגיל עם עד ל־8 ספרות
משמעותיות וללא ״זנב״ אפסים לאחר

הנקודה העשרונית. נקודה עשרונית ישר בהתחלה מלווה תמיד בְ־Ø; כך למשל Ø3. וְ־Ø.3.
Ø מודפס כספרה יחידה — Ø.
(iii) בטוי מחרוזת.
צרוף הסימולים במחרוזת־מורחבת; דרך אפשרית לכך היא בעזרת מרווחים לפניו או אחריו.
לתוי הבקרה מתקיימת השפעתם — בהתאם להגדרתם.
תוי הדפסה בלתי־מוכרים כגון — ?
(iv) AT m ,n
מוציאה כפלט תו בקרה של AT — שלאחריו בית עבור m (מספר השורה) ובית עבור n (מספר העמודה).
(v) TAB n
מוציאה כפלט תו בקרה של TAB כשלאחריו שני בתים עבור n (הפחות משמעותי — תחילה), שהיא מקום העצירה של TAB.
(vi) פרט צבע הוא בצורת הצהרה מבין PAPER, INK, FLASH, BRIGHT, INVERSE או OVER.

RANDOMIZE	מקביל לְ־RANDOMIZE Ø.
RANDOMIZE n	קובעת את משתנה המערך (הקרוי SEED = זרע) המשמש ליצירת הערך הבא של RND. באם Ø<>n, SEED מקבל את הערך n; באם Ø = n הוא מקבל את ערכו של משתנה מערכת אחר (הקרוי FRAMES) המונה את המסגרות שהוצגו עד כה במסך, וכך הוא אמור להיות אקראי למדי. RANDOMIZE מופיעה כְ־RAND על לוח המקשים. שגיאה B מתקבל באם n אינו בתחום Ø עד 65535.
READ V1 ,V2 ,... ,Vk	מגדירה את ערכי המשתנים בהתאם לבטויים שברשימת ה־DATA, לפי אותו הסדר. שגיאה C — באם בטוי הוא מסוג לא מתאים. שגיאה E — באם ישנם יותר משתנים מאשר בטויים ברשימת הַ־DATA.

REM...	ללא כל השפעה. הַ־'...' יכול להיות כל סדרת תוים שהיא מלבד ENTER. בסדרה זו יכולה להופיע־נקודותיים (:) ופסיקים (,) — כך שהצהרות אינן יכולות להופיע באותה שורה.
RESTORE	מקבילה לְ־RESTORE Ø.
RESTORE n	מחזירה את סמן ה־DATA להצהרת ה־DATA הראשונה בשורה שמספרה הינו לפחות n: הצהרת READ הבאה תתחיל משם.
RETURN	נוטלת יחוס להצהרה ממחסנית GO SUB וקופצת לאותה שורה. שגיאה 7 מתקיימת כאשר אין כל יחוס להצהרה במחסנית. קיימת שגיאה בתכנית שלך; ההצהרות GO SUB אינן מאוזנות עם הצהרות RETURN.
RUN	מקבילה לְ־RUN Ø.
RUN n	מבצעת CLEAR ואחר GO TO n.
SAVE f	שומרת את התכנית והמשתנים.
SAVE f LINE	שומרת את התכנית והמשתנים, כך שאם הם מוטענים למחשב — תתקיים קפיצה אוטומטית לתחילת התכנית והיא מתחילה ל"רוץ".
SAVE f LINE m	שומרת את התכנית והמשתנים שאם הם יוטענו למחשב — תתקיים קפיצה אוטומטית לשורה m.
SAVE f DATA ()	שומרת את המערך המספרי.
SAVE f DATA $ ()	שומרת את מערך התוים $.
SAVE f CODE m ,n SAVE f SCREEN $	שומרת n בתים החל מכתובת m. מקבילה לְ־SAVE f CODE 16384 ,6912. שומרת מידע על קסיטה, תוך הקניית השם f עבורה. שגיאה F באם f ריק או מכיל 11 או יותר תוים. ראה בפרק 20.

STOP	עוצרת את התכנית עם דווח 9. CONTINUE יחדש את מרוצת התכנית החל מההצהרה הבאה.
VERIFY	כמו במקרה של LOAD מלבד העובדה שהמידע אינו מוטען ל־RAM אלא מושווה עם מה שכבר קיים שם. שגיאה R באם אחת ההשוואות מלמדת כי קיים שוני בין בתים.

נספח D : תוכניות־דוגמא :

נספח זה מכיל כמה תוכניות — להדגמת יכולתו של ZX Spectrum.
הראשונה בהן דורשת בקלט את התאריך, ומדפיסה כתגובה את היום בשבוע המתאים לתאריך זה.

```
 10 REM convert date to day
 20 DIM d$(7,6): REM days of week
 30 FOR n=1 TO 7: READ d$(n): NEXT n
 40 DIM m(12): REM lengths of months
 50 FOR n=1 TO 12: READ m(n): NEXT n
100 REM input date
110 INPUT "day?";day
120 INPUT "month?";month
130 INPUT "year (20th century only)?";year
140 IF year<1901 THEN PRINT "20th century starts at 1901":
    GO TO 100
150 IF year>2000 THEN PRINT "20th century ends at 2000":
    GO TO 100
160 IF month<1 THEN GO TO 210
170 IF month>12 THEN GO TO 210
180 IF year/4-INT(year/4)=0 THEN LET m(2)=29: REM leap year
190 IF day>m(month) THEN PRINT "This month has only ";
    m(month);" days.": GO TO 500
200 IF day>0 THEN GO TO 300
210 PRINT "Stuff and nonsense. Give me a real date."
220 GO TO 500
300 REM convert date to number of days since start of century
310 LET y=year-1901
320 LET b=365*y+INT (y/4): REM number of days to start of year
330 FOR n=1 TO month-1: REM add on previous months
340 LET b=b+m(n): NEXT n
350 LET b=b+day
400 REM convert to day of week
410 LET b=b-7* INT (b/7)+1
420 PRINT day;"/";month;"/";year
430 FOR n=6 TO 3 STEP -1: REM remove trailing spaces
440 IF d$(b,n) <> " " THEN GO TO 460
450 NEXT n
460 LET e$=d$(b, TO n)
470 PRINT" is a "; e$; "day"
```

```
500 LET m(2)=28: REM restore February
510 INPUT "again?", a$
520 IF a$="n" THEN GO TO 540
530 IF a$ <> "N" THEN GO TO 100
1000 REM days of week
1010 DATA "Mon", "Tues", "Wednes"
1020 DATA "Thurs", "Fri", "Satur", "Sun"
1100 REM lengths of months
1110 DATA 31, 28, 31, 30, 31, 30
1120 DATA 31, 31, 30, 31, 30, 31
```

התכנית הבאה מטפלת ביארדים, רגלים ואינצ׳ים:

```
10 INPUT "yards?",yd,"feet?",ft, "inches?",in
40 GO SUB 2000: REM print the values
50 PRINT '"   =   ";
70 GO SUB 1000: REM the adjustment
80 GO SUB 2000: REM print the adjusted values
90 PRINT
100 GO TO 10
1000 REM subroutine to adjust yd, ft, in to the normal form for
     yards, feet and inches
1010 LET in=36*yd+12*ft+in: REM now everything is in inches
1030 LET s=SGN in: LET in=ABS in: REM we work with in
     positive, holding its sign in s
1060 LET ft=INT (in/12): LET in=(in-12*ft)*s: REM now in is ok
1080 LET yd=INT (ft/3)*s: LET ft=ft*s-3*yd: RETURN
2000 REM subroutine to print yd, ft and in
2010 PRINT yd;"yd";ft;"ft";in;"in";: RETURN
```

לפנינו תכנית המטילה מטבעות — עבור משחק סיני הקרוי 'I CHING':

```
5 RANDOMIZE
10 FOR m=1 TO 6: REM for 6 throws
20 LET c=0: REM initialize coin total to 0
30 FOR n=1 TO 3: REM for 3 coins
40 LET c=c+2+INT (2*RND)
50 NEXT n
60 PRINT "    ";
70 FOR n=1 TO 2: REM 1st for the thrown hexagram, 2nd for
   the changes
80 PRINT "——";
90 IF c=7 THEN PRINT "–";
100 IF c=8 THEN PRINT " ";
110 IF c=6 THEN PRINT "X";: LET c=7
120 IF c=9 THEN PRINT "0";: LET c=8
```



```
130 PRINT "——   ";
140 NEXT n
150 PRINT
160 INPUT a$
170 NEXT m: NEW
```

כדי להשתמש בה, הדפס והרץ אותה, ואחר לחץ על ENTER חמש פעמים כדי לקבל את שתי צורות הַ־"מגן־דוד". בחן בספר המתאר משחקים סינים — מהו הצעד הטוב ביותר שעליך לבצע. הקש ENTER בפעם השישית, והתכנית תמחוק את עצמה — בכדי למנוע ממך — מלהשתמש בה בקלות־ראש. אנשים רבים חשים כי הטקסטים הולמים יותר מאשר ציפיותיהם על בסיס אמונה — במזל; הדבר עשוי או עלול שלא להתקיים בְּ־ZX Spectrum. באופן כללי, מחשבים הנם יצורים חסרי כל אמונה. משחק נוסף לפנינו — משחק 'אוכל הנמלים' (pangolin). חשוב על חיה כלשהיא והמחשב ינסה לנחש מהי, בעזרת שאלות שהיא יציג לך — הנתנות לתשובה כן או לא. באם הוא לא שמע מעודו על החיה עליה חשבת, הוא יבקש ממך להציג שאלה בה יוכל הוא להשתמש בפעם הבא כדי לגלות באם מישהוא אחר יחשוב על אותה חיה בעתיד.

```
5 REM pangolins
10 LET nq=100: REM number of questions and animals
15 DIM q$(nq,50): DIM a(nq,2): DIM r$(1)
20 LET qf=8
30 FOR n=1 TO qf/2-1
40 READ q$(n): READ a(n,1): READ a(n,2)
50 NEXT n
60 FOR n=n TO qf-1
70 READ q$(n): NEXT n
100 REM start playing
110 PRINT "Think of an animal.","Press any key to continue."
120 PAUSE 0
130 LET c=1: REM start with 1st question
140 IF a(c,1)=0 THEN GO TO 300
150 LET p$=q$(c): GO SUB 910
160 PRINT "?": GO SUB 1000
170 LET in=1: IF r$="y" THEN GO TO 210
180 IF r$="Y" THEN GO TO 210
190 LET in=2: IF r$="n" THEN GO TO 210
200 IF r$<>"N" THEN GO TO 150
210 LET c=a(c,in): GO TO 140
300 REM animal
310 PRINT "Are you thinking of"
320 LET p$=q$(c): GO SUB 900: PRINT "?"
330 GO SUB 1000
340 IF r$="y" THEN GO TO 400
350 IF r$="Y" THEN GO TO 400
360 IF r$="n" THEN GO TO 500
370 IF r$="N" THEN GO TO 500
```

```
380 PRINT "Answer me properly when I'm","talking to you.": GO
    TO 300

400 REM guessed it
410 PRINT "I thought as much.": GO TO 800

500 REM new animal
510 IF qf>nq-1 THEN PRINT "I'm sure your animal is very",
    "interesting, but I don't have","room for it just now.": GO TO 800
520 LET q$(qf)=q$(c): REM move old animal
530 PRINT "What is it, then?": INPUT q$(qf+1)
540 PRINT "Tell me a question which dist-","inguishes
    between "
550 LET p$=q$(qf): GO SUB 900: PRINT " and"
560 LET p$=q$(qf+1): GO SUB 900: PRINT " "
570 INPUT s$: LET b=LEN s$
580 IF s$(b)="?" THEN LET b=b-1
590 LET q$(c)=s$(TO b): REM insert question
600 PRINT "What is the answer for"
610 LET p$=q$(qf+1): GO SUB 900: PRINT "?"
620 GO SUB 1000
630 LET in=1: LET io=2: REM answers for new and old animals
640 IF r$="y" THEN GO TO 700
650 IF r$="Y" THEN GO TO 700
660 LET in=2: LET io=1
670 IF r$="n" THEN GO TO 700
680 IF r$="N" THEN GO TO 700
690 PRINT "That's no good. ": GO TO 600

700 REM update answers
710 LET a(c,in)=qf+1: LET a(c,io)=qf
720 LET qf=qf+2: REM next free animal space
730 PRINT "That fooled me."

800 REM again?
810 PRINT "Do you want another go?": GO SUB 1000
820 IF r$="y" THEN GO TO 100
830 IF r$="Y" THEN GO TO 100
840 STOP

900 REM print without trailing spaces
905 PRINT " ";
910 FOR n=50 TO 1 STEP -1
920 IF p$(n)<>" " THEN GO TO 940
```



```
930 NEXT n
940 PRINT p$(TO n);: RETURN

1000 REM get reply
1010 INPUT r$: IF r$="" THEN RETURN
1020 LET r$=r$(1): RETURN

2000 REM initial animals
2010 DATA "Does it live in the sea",4,2
2020 DATA "Is it scaly",3,5
2030 DATA "Does it eat ants",6,7
2040 DATA "a whale", "a blancmange", "a pangolin", "an ant"
```

התכנית הבאה משרטטת את הדגל הבריטי:

```
  5 REM union flag
 10 LET r=2: LET w=7: LET b=1
 20 BORDER 0: PAPER b: INK w: CLS
 30 REM black in bottom of screen
 40 INVERSE 1
 50 FOR n=40 TO 0 STEP -8
 60 PLOT PAPER 0;7,n: DRAW PAPER 0;241,0
 70 NEXT n: INVERSE 0
100 REM draw in white parts
105 REM St. George
110 FOR n=0 TO 7
120 PLOT 104+n,175: DRAW 0,-35
130 PLOT 151-n,175: DRAW 0,-35
140 PLOT 151-n,48: DRAW 0,35
150 PLOT 104+n,48: DRAW 0,35
160 NEXT n
200 FOR n=0 TO 11
210 PLOT 0,139-n: DRAW 111,0
220 PLOT 255,139-n: DRAW -111,0
230 PLOT 255,84+n: DRAW -111,0
240 PLOT 0,84+n: DRAW 111,0
250 NEXT n
300 REM St. Andrew
310 FOR n=0 TO 35
320 PLOT 1+2*n,175-n: DRAW 32,0
330 PLOT 224-2*n,175-n: DRAW 16,0
340 PLOT 254-2*n,48+n: DRAW -32,0
350 PLOT 17+2*n,48+n: DRAW 16,0
360 NEXT n
370 FOR n=0 TO 19
```

```
380 PLOT 185+2*n,140+n: DRAW 32,0
390 PLOT 200+2*n,83-n: DRAW 16,0
400 PLOT 39-2*n,83-n: DRAW 32,0
410 PLOT 54-2*n,140+n: DRAW -16,0
420 NEXT n
425 REM fill in extra bits
430 FOR n=0 TO 15
440 PLOT 255,160+n: DRAW 2*n-30,0
450 PLOT 0,63-n: DRAW 31-2*n,0
460 NEXT n
470 FOR n=0 TO 7
480 PLOT 0,160+n: DRAW 14-2*n,0
485 PLOT 255,63-n: DRAW 2*n-15,0
490 NEXT n
500 REM red stripes
510 INVERSE 1
520 REM St. George
530 FOR n=96 TO 120 STEP 8
540 PLOT PAPER r;7,n: DRAW PAPER r;241,0
550 NEXT n
560 FOR n=112 TO 136 STEP 8
570 PLOT PAPER r;n,168: DRAW PAPER r;0,-113
580 NEXT n
600 REM St. Patrick
610 PLOT PAPER r;170,140: DRAW PAPER r;70,35
620 PLOT PAPER r;179,140: DRAW PAPER r;70,35
630 PLOT PAPER r;199,83: DRAW PAPER r;56,-28
640 PLOT PAPER r;184,83: DRAW PAPER r;70,-35
650 PLOT PAPER r;86,83: DRAW PAPER r;-70,-35
660 PLOT PAPER r;72,83: DRAW PAPER r;-70,-35
670 PLOT PAPER r;56,140: DRAW PAPER r;-56,28
680 PLOT PAPER r;71,140: DRAW PAPER r;-70,35
690 INVERSE 0: PAPER 0: INK 7
```

באם אינך בריטי, נסה לצייר את דגלך. דגלים בעלי 3 פסים בצבעים שונים קלים מאד לביצוע, למרות שכמה מן הצבעים — כגון הכתום בדגל האירי — עלולים לסבך מעט את התכנית. באם אתה אמריקאי, תוכל להעזר בתו *.

לפנינו תכנית — 'התלין' (hangman) בה שחקן אחד אחד מכניס מילה והשני מנסה לנחש מהי.

```
5 REM Hangman
10 REM set up screen
20 INK 0: PAPER 7: CLS
30 LET x=240: GO SUB 1000: REM draw man
```



```
40 PLOT 238,128: DRAW 4,0: REM mouth
100 REM set up word
110 INPUT w$: REM word to guess
120 LET b=LEN w$: LET v$=" "
130 FOR n=2 TO b: LET v$=v$+" "
140 NEXT n: REM v$=word guessed so far
150 LET c=0: LET d=0: REM guess & mistake counts
160 FOR n=0 TO b-1
170 PRINT AT 20,n;"-";
180 NEXT n: REM write -'s instead of letters
200 INPUT "Guess a letter: ";g$
210 IF g$="" THEN GO TO 200
220 LET g$=g$(1): REM 1st letter only
230 PRINT AT 0,c;g$
240 LET c=c+1: LET u$=v$
250 FOR n=1 TO b: REM update guessed word
260 IF w$(n)=g$ THEN LET v$(n)=g$
270 NEXT n
280 PRINT AT 19,0;v$
290 IF v$=w$ THEN GO TO 500: REM word guessed
300 IF v$<>u$ THEN GO TO 200: REM guess was right
400 REM draw next part of gallows
410 IF d=8 THEN GO TO 600: REM hanged
420 LET d=d+1
430 READ x0,y0,x,y
440 PLOT x0,y0: DRAW x,y
450 GO TO 200
500 REM free man
510 OVER 1: REM rub out man
520 LET x=240: GO SUB 1000
530 PLOT 238,128: DRAW 4,0: REM mouth
540 OVER 0: REM redraw man
550 LET x=146: GO SUB 1000
560 PLOT 143,129: DRAW 6,0, PI/2: REM smile
570 GO TO 800
600 REM hang man
610 OVER 1: REM rub out floor
620 PLOT 255,65: DRAW -48,0
630 DRAW 0,-48: REM open trapdoor
640 PLOT 238,128: DRAW 4,0: REM rub out mouth
650 REM move limbs
655 REM arms
660 PLOT 255,117: DRAW -15,-15: DRAW -15,15
670 OVER 0
680 PLOT 236,81: DRAW 4,21: DRAW 4,-21
```

```
690 OVER 1: REM legs
700 PLOT 255,66: DRAW -15,15: DRAW -15,-15
710 OVER 0
720 PLOT 236,60: DRAW 4,21: DRAW 4,-21
730 PLOT 237,127: DRAW 6,0,-PI/2: REM frown
740 PRINT AT 19,0;w$
800 INPUT "again? ";a$
810 IF a$="" THEN GO TO 850
820 LET a$=a$(1)
830 IF a$="n" THEN STOP
840 IF a$(1)="N" THEN STOP
850 RESTORE : GO TO 5
1000 REM draw man at column x
1010 REM head
1020 CIRCLE x,132,8
1030 PLOT x+4,134: PLOT x-4,134: PLOT x,131
1040 REM body
1050 PLOT x,123: DRAW 0,-20
1055 PLOT x,101: DRAW 0,-19
1060 REM legs
1070 PLOT x-15,66: DRAW 15,15: DRAW 15,-15
1080 REM arms
1090 PLOT x-15,117: DRAW 15,-15: DRAW 15,15
1100 RETURN
2000 DATA 120,65,135,0,184,65,0,91
2010 DATA 168,65,16,16,184,81,16,-16
2020 DATA 184,156,68,0,184,140,16,16
2030 DATA 204,156,-20,-20,240,156,0,-16
```


נספח E: מספרים בינאריים והקסדצימליים:

נספח זה מתאר כיצד מחשבים־מונים, בעזרת המערכת הבינארית. רוב השפות — מונות באופן פחות או יותר זהה — בעזרת תבניות של 10; כך לדוגמא מתקיים —

עשרים, עשרים ואחד, עשרים ושניים, ..., עשרים ותשע שלושים, שלושים ואחד, שלושים ושניים, ..., שלושים ותשע ארבעים, ארבעים ואחד, ארבעים ושניים, ..., ארבעים ותשע.

וכך הלאה. בכל אופן, הסיבה היחידה לשימוש ב־10 כתבנית קשורה בעובדה שבידינו 10 אצבעות. במקום להשתמש במערכת העשרונית — שבסיסה 10, המחשבים משתמשים בצורה בינארית מסוימת הנקראת הקסאדצימלית — שבסיסה 16. מאחר ובבסיס העשרוני מוגדרות רק 10 ספרות, עלינו להגדיר למערכת ההקסאדצימלית 6 ספרות נוספות. נשתמש באותיות A, B, C, D, E וְ־F. מה קיים לאחר F? כשם שאנו, בעזרת עשרת אצבעותינו, רושמים 1Ø במשמעות עשר — המחשבים רושמים גם הם 1Ø אלא במשמעות שש־עשרה. המערכת המספרית שלהם הינה:

המספר:	יצוג הקסדצימלי:
אפס	Ø
אחד	1
שנים	2
⋮	⋮
תשע	9
⋮	

בדומה למערכת לה אנו רגילים, אך בהמשך:

עשר	A
אחד־עשר	B
שנים־עשר	C
שלושה־עשר	D
ארבעה־עשר	E
חמשה־עשר	F
ששה־עשר	1Ø
שבעה־עשר	11
⋮	⋮
עשרים וחמש	19
עשרים ושש	1A
עשרים ושבע	1B
⋮	⋮
עשרים ואחד	1F
שלושים ושנים	2Ø

שלושים ושלוש	21
:	:
מאה חמישים ושמונה	9E
מאה חמישים ותשע	9F
מאה ושישים	AØ
מאה ששים ואחד	A1
:	:
מאה ושמונים	B4
	:
מאתיים חמישים וארבע	FE
מאתיים חמישים וחמש	FF
מאתיים חמישים ושש	1ØØ

באם אתה משתמש ברישום הקסדצימלי וברצונך להגדיר בפשטות עובדה זו, רשום h — בסופו של כל מספר. לדוגמא־עבור — מאה חמישים ושמונה בהקסדצימלי רשום: 9E h.

לבטח אינך רואה מה הקשר של הרישום ההקסדצימלי למחשבים. למעשה, מחשבים מתנהגים כאילו היו ברשותם רק שתי ספרות — המיוצגות ע״י מתח־נמוך או off — לא פעיל (Ø), ומתח־גבוה או on־פעיל (1). זוהי המערכת הבינארית, ושתי ספרות הבינאריות קרויות ביטים (bits): אם־כך, ביט הוא Ø או 1.

במערכות השונות מתקיים:

המספר	יצוג־עשרוני	יצוג הקסדצימלי	יצוג בינארי
אפס	Ø	Ø	ØØØØ או Ø
אחד	1	1	ØØØ1 או 1
שנים	2	2	ØØ1Ø או 1Ø
שלושה	3	3	ØØ11 או 11
ארבעה	4	4	Ø1ØØ או 1ØØ
חמשה	5	5	Ø1Ø1 או 1Ø1
ששה	6	6	Ø11Ø או 11Ø
שבעה	7	7	Ø111 או 111
שמונה	8	8	1ØØØ
תשעה	9	9	1ØØ1
עשרה	1Ø	A	1Ø1Ø
אחד־עשר	11	B	1Ø11
שנים־עשר	12	C	11ØØ
שלושה־עשר	13	D	11Ø1
ארבעה־עשר	14	E	111Ø
חמשה־עשר	15	F	1111
ששה־עשר	16	1Ø	1ØØØØ

הנקודה החשובה היא שש־עשרה שווה לשנים בחזקת ארבע, והיא מפשטת את המעבר מהבסיס הבינארי לבסיס ההקסדצימלי ולהפך־בקלות רבה.

כדי לתרגם מספר הקסדצימלי לבינארי, החלף כל ספרה הקסדצימלית במספר הבינארי המתאים — בן ארבעה ביטים, בעזרת הטבלה שלמעלה.

כדי לתרגם מספר בינארי להקסדצימלי, חלק את המספר הבינארי לקבוצות בנות 4 ביטים, החל מצדו הימני של המספר ואחר־תרגם כל קבוצה לספרה ההקסדצימלית המתאימה.

מסיבה זו, למרות שלמעשה המחשבים עצמם מדברים בשפה הבינארית, אנשים רושמים בדרך כלל את המספרים האגורים בזכרון ברישום הקסדצימלי.

הביטים מקובצים, בדרך כלל, בקבוצות בנות 8 ביטים, הקרויו בתים (Bytes). בית בודד יכול לייצג מספר מ־Ø וד מאתים חמישים וחמש (11111111 בבינארי או FF בהקסדצימלי), או לחלופין כל תו מאוסף התוים של — ZX Spectrum. ערכו נתן לרשום בעזרת שתי ספרות הקסדצימליות.

נתן לחבר 2 בתים יחד כדי ליצור — "מילה" (Word). המילה מורכבת מ־16 ביטים או 4 ספרות הקסדצימליות והיא מייצגת מספר בתחום Ø עד המספר העשרוני $65535 = 1 - 2^{16}$.

בית כולל תמיד 8 ביטים אולם מילה יכולה להיות באורך שונה — עבור מחשבים שונים.

הסימון BIN שהוזכר בפרק 14 הוא אמצעי לרישום מספרים בקוד בינארי ב־Zx Spectrum: 'BIN Ø' מייצג אפס, 'BIN 1' מייצג 1, 'BIN 1Ø' מייצג 2 בבסיס עשר, וכך הלאה.

אתה רשאי להשתמש רק ב־Ø או 1 כך שהמספר חייב להיות מספר שלם־חיובי; כדוגמא, אינך יכול לרשום "BIN־11" עבור הערך העשרוני 3 —, עליך לרשום במקום זאת 'BIN 11—'. המספר חייב להיות קטן מהמספר העשרוני 65535, דהיינו אינו יכול להכיל יותר מ־16 ביטים.

ATTR הינה למעשה בינארית. באם תתרגם את התוצאה מ־ATTR לקוד בינארי תוכל לרשמה בשמונה ביטים.

הראשון הינו 1 עבור הבהוב וØ עבור יציב (חוסר־הבהוב) השני הינו 1 עבור בהירות גבוהה ו־Ø לבהירות רגילה. השלושה הבאים בתור הם הקוד הבינארי של צבע ה'ניר' השלושה האחרונים הם הקוד הבינארי של צבע ה'דיו'.

קודי־הצבעים מוגדרים גם הם בקודים בינאריים: כל קוד, כשהוא מופיע בערכו הבינארי, מורכב משלושה ביטים שהראשון בהם מייצג את הצבע הירוק, השני את האדום והשלישי את הכחול.

שחור אינו מכיל צבעים כלשהם ולכן כל הביטים בקוד המתאים לו הם Ø (off — לא פעילים) — והקוד הוא ØØØ.

הצבעים הבסיסיים — ירוק אדום וכחול כוללים רק ביט אחד שערכו 1 (on־פעיל) מבין השלושה. קודיהם הנם 1ØØ, Ø1Ø, ו־ØØ1 בבינארי או בהתאמה 4, 2, ו־1 בבסיס עשר.

שאר הצבעים הנם תוצאה מערבוב צבעי יסוד, ולכן הם מכילים שני ביטים בערך 1 או אף שלושה.